HOCHSCHULE
MITTWEIDA
UNIVERSITY OF
APPLIED SCIENCES

BACHELORARBEIT

Herr

Andreas Rieger

Steigerung der Usability und Informationstransparenz in IBM Connections durch die Umsetzung von Konzepten zur Verknüpfung von Inhalten mit prototypischer Umsetzung

Mittweida, 2011

Fakultät Mathematik, Naturwissenschaften, Informatik

BACHELORARBEIT

Steigerung der Usability und Informationstransparenz in IBM Connections durch die Umsetzung von Konzepten zur Verknüpfung von Inhalten mit prototypischer Umsetzung

Autor:

Herr

Andreas Rieger

Studiengang:

Wirtschaftsinformatik

Seminargruppe:

WF07w1-B

Erstprüfer:

Prof. Wilfried Schubert

Zweitprüfer:

Dipl.-Informatiker Tino Schmidt

Einreichung:

Mittweida, 04.10.2011

Verteidigung/Bewertung:

Mittweida, 2011

Bibliografische Beschreibung:

Rieger, Andreas:
Steigerung der Usability und Informationstransparenz in IBM Connections
durch die Umsetzung von Konzepten zur Verknüpfung von Inhalten mit prototypischer Umsetzung. - 2011. - 75 S.
Mittweida, Hochschule Mittweida, Fakultät Mathematik, Naturwissenschaften, Informatik, Bachelorarbeit, 2011

Referat:

Die Arbeit beschreibt wie in IBM Connections Erweiterungen integriert werden können, um die Informationstransparenz zu steigern. Dabei wird die Entwicklung eines Prototyps beschrieben, der die zuvor analysierten Programmschnittstellen nutzt. Der Prototyp wird verwendet um fehlende Funktionen zu ergänzen.

i Inhaltsverzeichnis

ii Abbildungsverzeichnis

iii Tabellenverzeichnis

iv Quellcode-Verzeichnis

v Abkürzungsverzeichnis

AJAX

Asynchronous JavaScript and XML

API

Application Programming Interface, Programmierschnittstelle

DOM

Document Object Model

EA

Enterprise-Application, Unternehmensanwendung

HTML

Hypertext Markup Language

IBM

International Business Machines Corporation

JAAS

Java Authentication and Authorization Service

JS

JavaScript

JSON

JavaScript Object Notation

JSP

JavaServerPage

JSTL

Java Standard Tag Library

JVM

Java Virtual Machine

MVC

Modell-View-Controll

vCard

Versit-Card, elektronische Visitenkarte

XML

Extensible Markup Language

vi Glossar

Dojo Toolkit (kurz: Dojo)

Das Dojo Toolkit ist eine freie JavaScript-Bibliothek, die die Bearbeitung des DOM vereinfacht und die Verwendung von AJAX erleichtert. Dojo wird von IBM für eine Vielzahl von Webanwendungen genutzt. [1]

iWidgets

iWidgets sind Widgets die nach der iWidgets-Spezifikation von IBM erstellt wurden. iWidgets werden verwendet um verschiedenste IBM Software-Produkte zu erweitern, dazu zählen IBM Connections [2], IBM Mashup Center [3] und IBM WebSphere Portal [4].

JQuery

JQuery ist eine freie JavaScript-Klassenbibliothek, die es vereinfacht HTML-Teile (das DOM) zu bearbeiten und zwischen ihnen zu navigieren. [5]

Language-Properties

Language-Properties werden in einem System verwendet, um eine Internationalisierung für eine Anwendung zu realisieren. Die Dateien bestehen aus MessageKeys und der dazugehörigen Übersetzung. Für jedes unterstütztes Land und deren Sprache muss dann eine Property-Datei gepflegt werden. Die Property-Daten werden als sogenanntes Ressource-Bundle zusammengefasst, welches dann je nach übergebener Sprache, die richtige Übersetzung zurückliefert, wenn diese vorhanden ist. Zusätzlich gibt es als Fallback noch eine default-Datei, die meist mit der englischen identisch ist. [6]

Lotus Quickr

Lotus Quickr ist Team-Collaboration-Software von IBM mit dem Schwerpunkt Dokumentenmanagement. Diesen gibt es in zwei Ausführungen, die sich in der grundliegenden Technik unterscheidet. Eine Version basiert auf den WebSphere Application Server und WebSphere Portal, die andere Version auf den Lotus Domino Server. [7]

WebSphere Application Server

Der WebSphere Application Server ist eine Java Application Server von IBM, der zusätzlich zu den Standardaufgaben eines Application-Servers eine Reihe von Zusatzdiensten wie die Nutzerverwaltung bietet. Die aktuelle Version ist 7.0. [8]

Widgets

Widgets sind kleine Programmteile, die in ein anderes Programm eingebunden werden können. Sie erweitern damit die Nutzbarkeit der Seite, in der diese verwendet werden oder stellen externe Inhalte bereit. Widget-Spezifikationen gibt es von vielen großen Softwareherstellern. [9]

vii Hinweis zur Sprachwahl

Communities vs. Communitys

Communities steht immer für die englische Bezeichnung der Anwendung. Im deutschen wird die Anwendung immer mit Communitys bezeichnet. Auch das Objekt Community wird im allgemeinen Zusammenhang mit dem deutschen Plural, im informatischen Zusammenhang mit dem englischen Plural verwendet. Dies betrifft auch andere Anwendungen wie zum Beispiel Activities und Aktivitäten.

1 Einleitung

In vielen mittelständischen und großen Unternehmen spielt die Zusammenarbeit und Kommunikation der Mitarbeiter untereinander eine wichtige Rolle auch über Organisationsgrenzen hinweg. Zusätzlich ist es von bedeutender Wichtigkeit Wissen und Informationen innerhalb des Unternehmens gemeinsam zu nutzen und großflächig an verschiedenen Standorten transparent zur Verfügung zu stellen. Dabei liefert ein Produkt der IBM Corporation einen entscheidenden Beitrag für verschiedenste Organisationen zur Sozialen Zusammenarbeit. Gemeint ist IBM Connections als „Social Software für Unternehmen“.

Es ermöglicht es gemeinsam Aufgaben innovativ und schnell zu lösen und gewonnene Erkenntnisse untereinander zu teilen. Doch in großen Unternehmen mit einer Vielzahl von Personen und internen Gruppierungen kann ein Mitarbeiter schnell der Überblick über einzelne Bereiche verlieren und so den Mehrwert des Produkts als schnelle und komfortable Wissensbasis nicht Nutzen.

1.1 Problembeschreibung

IBM bietet für einen Teil der in Connections angebotenen Softwarekomponenten keine schnelle Überblicksmöglichkeit wie es von anderen ähnlichen Komponenten in Connections angeboten wird. Informationen dieser Funktionen lassen sich in den Überblicksseiten der Software nicht oder nur äußerst umständlich in eine optimierte und vor allem gefilterte Ansicht bringen. Dieses Defizit wurde von Partner IBMs und Kunden der Communardo Software GmbH beobachtet und kritisiert.

1.2 Ziele

In der Bachelorarbeit soll untersucht werden, welche Anzeigemöglichkeiten die Überblickseiten von IBM Connections bietet und wie diese erweitert werden können. Dabei soll analysiert werden, welche Schnittstellen die Software bietet, die für eine sinnvolle Ergänzung vorhanden sind. Im Anschluss sollen die Erkenntnisse in die Entwicklung eines Prototypens münden, der die Mängel der Standardsoftware behebt oder zumindest abschwächt. Es sollen Erweiterungen für eine individuelle Nutzung der Überblickseiten angelegt werden, mit deren Hilfe der Nutzer entscheiden kann, welche Inhalte er angezeigt bekommt und welche nicht.

1.3 Abgrenzung

Die Arbeit soll nicht analysieren welche Erweiterungen sinnvoll oder nötig sind, da dies als vorgegeben gesetzt wird. Es wird eine allgemeine Detailstufe für die Filtermöglichkeiten der einzelnen Anwendungen verwendet, die nicht zu speziell auf einzelne Anwendungen konzipiert werden.

Um der Bearbeitungszeit der Bachelorarbeit gerecht zu werden, wird darauf verzichtet den kompletten Softwareentwicklungsprozess abzubilden. Es werden lediglich die wichtigsten Punkte beschrieben.

1.4 Aufbau der Arbeit

Für das bessere Verständnis der Arbeit soll in diesem Abschnitt der Aufbau der Arbeit durch eine kurze inhaltliche Beschreibung der einzelnen Kapitel im Gesamtzusammenhang erfolgen. Kapitel 2 dient zur Erklärung was IBM Connections ist und welche Grundlegenden Funktionen es bietet. Außerdem soll gezeigt werden, wie die einzelnen Komponenten und das gesamte Softwareprodukt aufgebaut und mit welchen anderen Anwendungen in der Standardverwendung kommuniziert wird. Im dritten Kapitel wird untersucht, welche Überblicksseiten in IBM Connections enthalten sind und was darauf angezeigt werden kann. Ebenfalls soll analysiert werden, wie IBM diese Ansichten realisiert hat. Zusätzlich soll gezeigt werden, welche Erweiterungsmöglichkeiten IBM für das gegebene Problem vorgesehen hat und wie diese Schnittstellen aufgebaut und genutzt werden können. Kapitel 4 befasst sich mit der Realisierung des Prototyps. Es wird beschrieben welche Techniken eingesetzt wurden. Des Weiteren soll dabei auf wichtige Implementierungsdetails eingegangen. Im fünften und letzten Kapitel werden die erzielten Ergebnisse untersucht und Schlussfolgerungen auf mögliche Verbesserungen oder Änderungen geben.

2 IBM Connections

IBM Connections ist ein Softwareprodukt von IBM, welches in den Bereich „Social Software“ bzw. „Social Enterprise“ zählt. IBM Connections dient dazu Nutzer und Information miteinander zu verknüpfen und so im Unternehmen für jedermann leicht erreichbar zu machen. Es hilft dabei Daten innerhalb von Teams oder Projekträumen, sogenannten Communitys, zu verwalten, zu organisieren und zu erweitern. Zusätzlich gibt es die Möglichkeit im Wissensnetzwerk nach Personen und Informationen zu suchen, um so die tägliche Arbeit besser zu bewerkstelligen und besser miteinander zu arbeiten. [10]

2.1 Aufbau von IBM Connections

Das System besteht aus einem Server mit installierten WebSphere Application Server. Dieser Server dient als Grundlage für IBM Connections. Zusätzlich zu diesem Server wird ein Datenbanksystem benötigt, wahlweise IBM DB2, Oracle Database oder Microsoft SQL [11 S. Databases]. Naheliegend ist hier die Verwendung des IBM DB2 Servers. Zusätzlich zum Datenbanksystem wird ein Verzeichnisdienst zur Nutzer- und Gruppenverwaltung benötigt. IBM Connections ist hier mit einer Vielzahl von Produkten kompatibel (siehe Tabelle 1). Als kleine, leistungsstarke Lösung wurde auf den IBM Tivoli Directory Server gesetzt. Um das Senden von Benachrichtigungen zu ermöglichen muss ein Mail-Server angebunden werden. Auf diesem kann verzichtet werden. Es sollte im produktiven System verwendet werden, um die gesamte Möglichkeit von IBM Connections zu erhalten. Im Test- und Entwicklungssystem wurde ein Postfix Message Transport Agent verwendet.

Tabelle 1 unterstützte Verzeichnisdienste

Unterstütze Verzeichnisdienste [11 S. LDAP Servers]
Lotus Domino 8.0.2
Microsoft Active Directory 2003 SP2
Microsoft Active Directory 2003 ADAM
Microsoft Active Directory 2008
Novell eDirectory 8.8 SP5
Sun Java System Directory Server 6.3
Tivoli Directory Server 6.2 FP2

(SP: Service Pack, FP: Fix Pack, ADAM: Active Directory Application Mode)

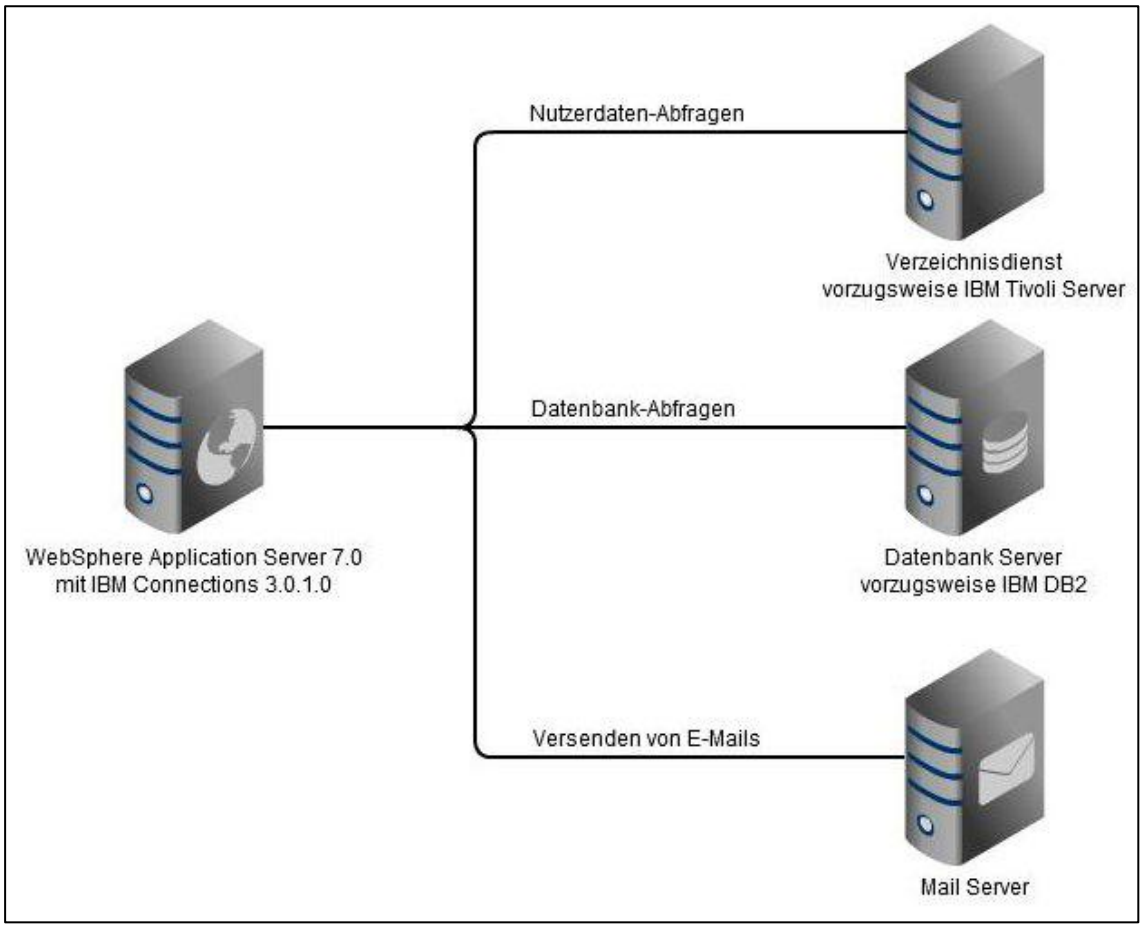


Abbildung 1 Deployment-Diagramm

2.2 Enterprise-Anwendungen (EA)

Um eine möglichst große Flexibilität zu erreichen, besteht IBM Connections aus mehreren kleineren Enterprise-Anwendungen, die beliebig je nach Anwendungsfall installiert und genutzt werden können. Jede EA ist ein eigenständiges Enterprise Application Archive [12 S. 149].

Es gibt drei zentrale EAs, die installiert werden müssen. Diese sind „Homepage“, „Search“ und „News“ [13 S. "Procedure #10"]. Damit dienen sie als Grundgerüst von IBM Connections. Andere Enterprise-Anwendungen verwenden Funktionalität dieser zentralen Komponenten, wie die Suchfunktionalität.

Jede dieser Anwendungen hat eine eigene Datenbankanbindung und in der Regel eine eigene Datenbank. Lediglich „News“, „Search“ und „Homepage“ teilen sich eine Datenbank [14 S. "Procedure #6c"]. Zusätzlich besitzt jede Anwendung, außer „Homepage“, eine Anbindung an den Mail-Server um Benachrichtigungen zu versenden.

2.3 Verwendung von IBM Connections

2.3.1 Verwendung der Einzelteile

2.3.1.1 Communitys

Communitys sind Gemeinschaften von Nutzer, die zusammen ein Thema bearbeiten oder sonstige Interessen teilen. Communitys können dabei verschiedene reale Personengruppen, wie Projektteams oder Interessengemeinschaften, widerspiegeln. Zusätzlich können SubCommunitys zur besseren Organisation verwendet werden. SubCommunitys sind dabei die Möglichkeit Communitys innerhalb einer Community anzulegen. Sie bieten die gleichen Optionen, wie eine Community, Ausnahme ist hierbei das Anlegen weiterer SubCommunitys. Somit ist es möglich Communitys hierarchisch auf zwei Ebenen anzuordnen. Dadurch wird es einfach sich in der Vielzahl der entstehenden Communitys zurechtzufinden (siehe Abbildung 2). [15]

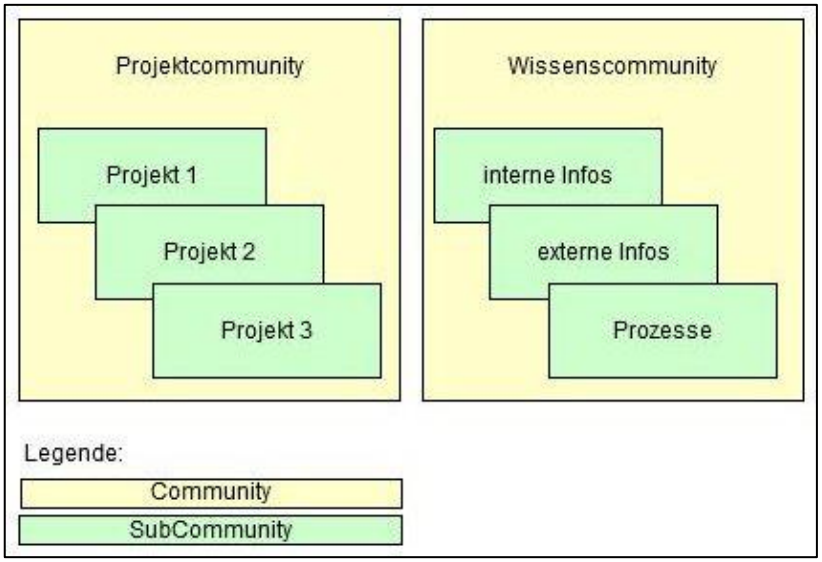


Abbildung 2 Communitys als Organisationsstruktur

Communitys bieten jedoch nur einen Mehrwert, wenn diese durch Anwendungen mit Inhalt gefüllt werden. Anwendungen sind Blogs, Wikis, Nachrichten-Feeds, Foren, Lesezeichen, Dateien, Aktivitäten, Ideation-Blog und Mediengalerie. Bis auf Nachrichten-Feeds, Ideation-Blog [16] und Mediengalerie [17] sind alle Anwendungen auch ohne eine Community möglich, sodass sich jeder Nutzer einen eigenen Blog anlegen kann, ohne dafür Mitglied einer Community zu sein.

Die Anwendung Communitys wird durch die EA „Communities“ realisiert. Die EA „Communities“ ist eine der größten Anwendungen innerhalb von IBM Connections. Sie besitzt eine Anbindung an den Mail-Server und an eine Datenbank mittels eigener JAAS-

Implementierung [14 S. "Procedure #6c"]. Zusätzlich sind in ihr die Web-Apps „lc.widget.web“, „Community Blogs Widget“, „Ideation Blog Widget“ und „lc.widgets.ActivityList“ enthalten, welche eine Schnittstelle zu anderen EA mit Hilfe von Widgets herstellt.

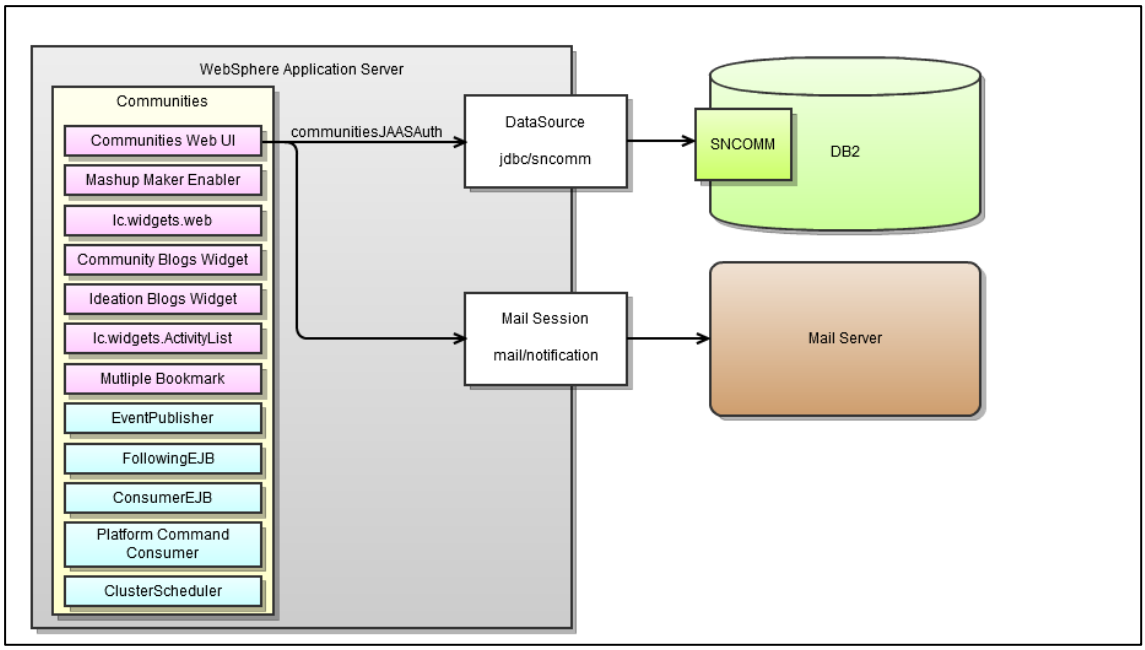


Abbildung 3 Aufbau der Communities.ear

2.3.1.2 Blogs, Ideation-Blogs und Aktivitäten

Blogs bietet die Möglichkeit einen Blog in eine Community einzubinden, in denen Sachverhalte protokolliert werden können. Anderen Nutzern ist es auf Wunsch gestattet, dies zu kommentieren. Eine Erweiterung der Blogs-Anwendung ist der Ideation-Blog [16]. Dieser wird verwendet um Ideen zu sammeln, diese zu kommentieren und zu bewerten. Es bietet sich anschließend die Möglichkeit eine Idee direkt in eine Aktivität bzw. Aufgabe umzuwandeln [18 S. "Procdure #6"]. Mit Hilfe von Aktivitäten können Arbeitsabläufe organisiert werden. Es gibt die Möglichkeit E-Mails oder andere Objekte mit einer Aktivität zu verknüpfen. Dies ermöglicht eine einfachere Bearbeitung, da Inhalte direkt vorhanden sind.

Die Anwendung Blogs und damit auch die Spezialisierung Ideation-Blog ist in „Blogs.ear“ enthalten. Die EA „Blogs“ besitzt auch wie die EA „Activities“ eine Anbindung an den Mail-Server und an eine Datenbank, die mittels „blogsJAASAuth“ hergestellt wird [14 S. "Procedure #6c"]. „blogsJAASAuth“ nutzt dabei die JAAS-API.

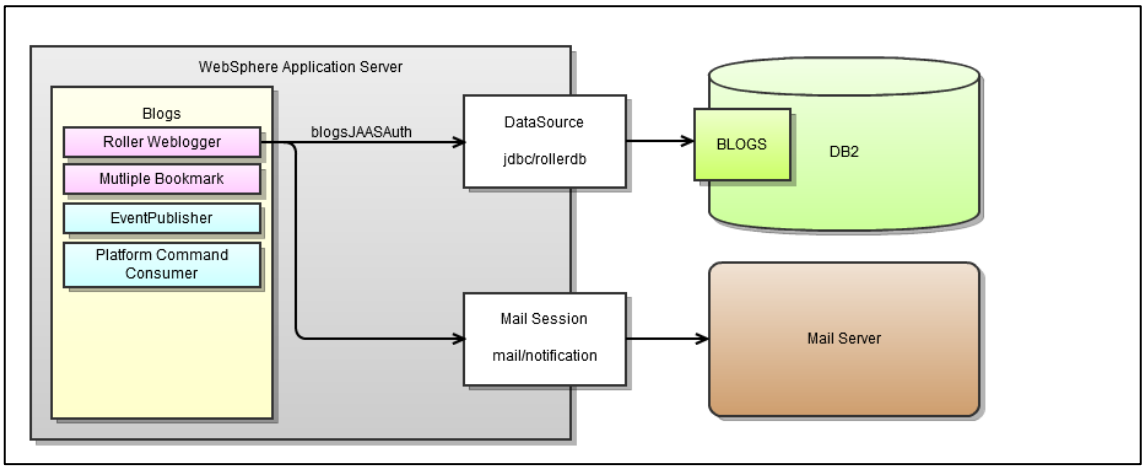


Abbildung 4 Aufbau der Blogs.ear

Die EA, die Activities umsetzt, besitzt eine Anbindung zu einer Datenbank, die mithilfe der „activitiesJAASAuth“ realisiert wird [14 S. "Procedure #6c"]. Dies ist eine Implementierung des JAAS, die die Anmeldung am Datenbanksystem sicherstellt. Zusätzlich gibt es eine Anbindung an den Mail-Server. Als weitere nennenswerte Komponenten gibt es den „Quickr_Document_Picker“, der es ermöglich Dateien oder E-Mails direkt an eine Aktivität zu binden. Zu finden ist die EA in der „Activities.ear“

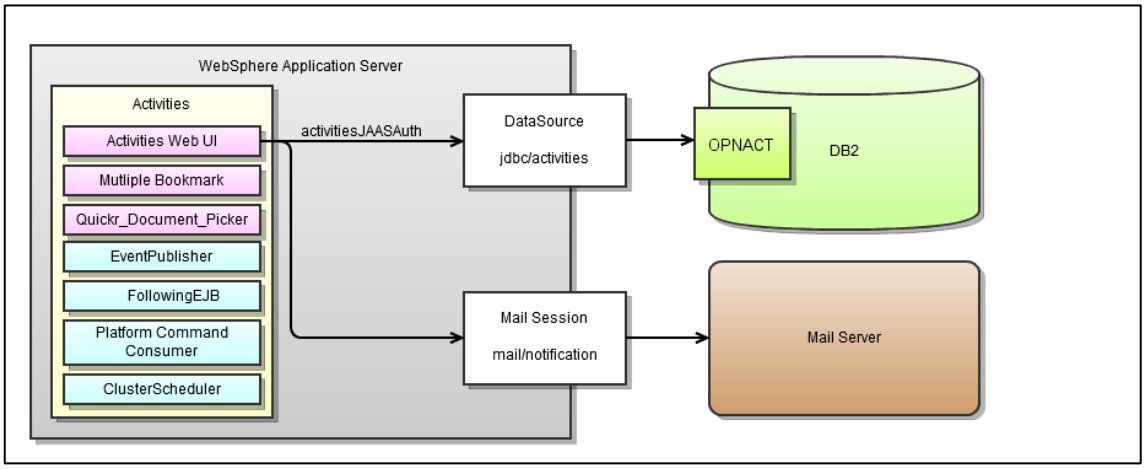


Abbildung 5 Aufbau der Activities.ear

2.3.1.3 Wikis und Forum

Durch Wikis kann Wissen innerhalb einer Community besser getauscht und erweitert werden. Es lassen sich Seiten zu Themen erstellen, die dann von der Gemeinschaft genutzt, verändert und erweitert werden können, um dadurch ein Höchstmaß an Wissenstransparenz zu ermöglichen. [19] Unklarheit können direkt innerhalb eines Forums geklärt werden. So entstehende Diskussionsrunden ermöglichen eine bessere und schnellere Zusammenarbeit auch über lokale Beschränkungen hinweg. [20]

Die EA „Wikis“ nutzt auch wie die anderen EAs eine Datenbankverbindung und eine eigene Mail-Session. Bei ihr ist der Ursprung aus Quickr zu erkennen („qkr“ steht auch hier für Quickr).

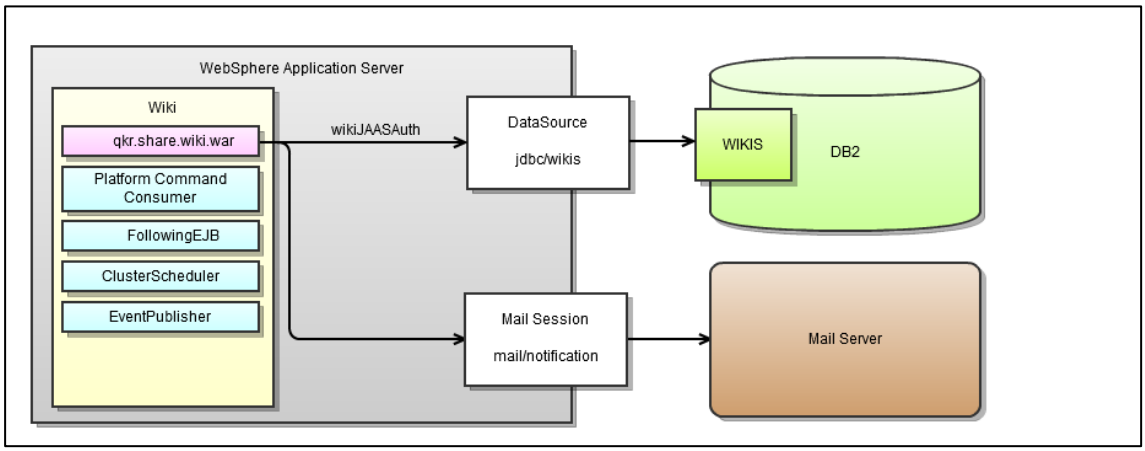


Abbildung 6 Aufbau der Wiki.ear

Die EA „Forum“ ist eine der neueren Anwendungen. Sie wurde erst in IBM Connections 3.0 eingeführt [21 S. "What's new in using?"] und besitzt wie die meisten Anwendungen eine Datenbank- und Mail-Server-Anbindung. Zusätzlich liefert das EJB-Modul (ClusterScheduler) die Synchronisation im Clusterbetrieb.

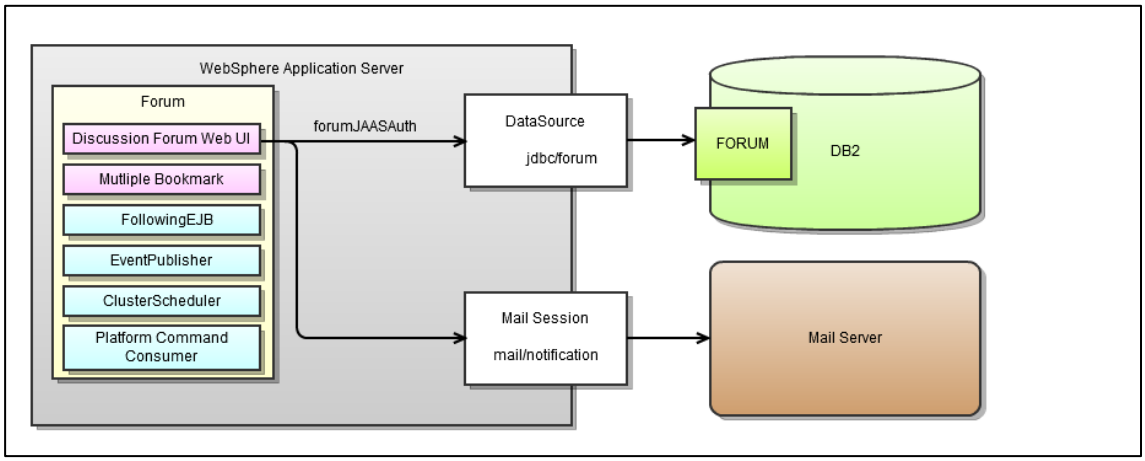


Abbildung 7 Aufbau der Forum.ear

2.3.1.4 Lesezeichen und Feeds

Die Integration von Lesezeichen und Nachrichten-Feeds bietet den Nutzern die Möglichkeit, wichtige oder interessante externe Inhalte einzubinden und diese mit der Gemeinschaft zu teilen. Die Funktion der Lesezeichen ist teilweise auch in anderen Anwendungen als „Links“ integriert um anwendungsweit eine Lesezeichen-Sammlung zu etablieren.

Die EA „Dogear“ ist eine einfache EA, die lediglich eine Datenbank- und Mail-Server-Anbindung besitzt. Sie ist für die Verwaltung der Lesezeichen zuständig. Im Laufe der Connections-Entwicklung hat sich der Name von „Dogear“ in „Bookmarks“ (Lesezeichen) geändert [22].

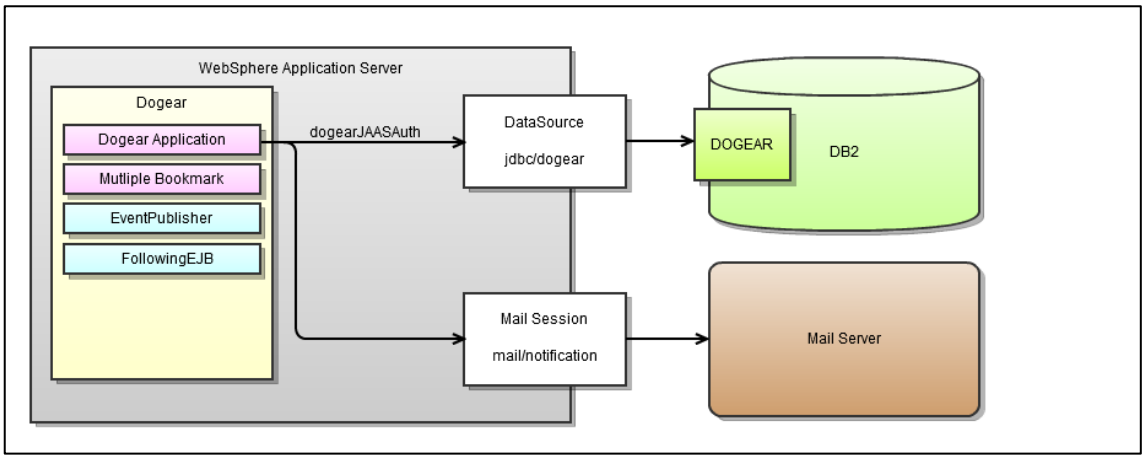


Abbildung 8 Aufbau der Dogear.ear

2.3.1.5 Dateien und Mediengalerie

Zusätzlich gibt es die Möglichkeit Dateien zur Verfügung zu stellen. Diese können direkt für eine Community hochgeladen oder aus bereits bestehenden Dateisammlungen freigeben werden. Damit ist es möglich ein und dasselbe Dokument an mehrere Communitys zu verteilen. Die Mediengalerie bietet zusätzlich noch die Möglichkeit hochgeladene Multimedia-Inhalte direkt in einem Widget anzuzeigen. Dies umfasst bisher Fotos und Videos [17].

Die Dateien und die Mediengalerie wird durch die EA „Files“ realisiert. Die EA „Files“ ist eine der ältesten Anwendungen in IBM Connections. Sie basiert ursprünglich auf die Quickr-Files-Anwendung, was noch am Namen erkennbar ist („qkr“ steht für Quickr). Diese Anwendung besitzt eine Datenbank- und Mail-Server-Anbindung. Zusätzlich ist ein Cache angebunden, der die hochzuladenden Dateien zwischenspeichert.

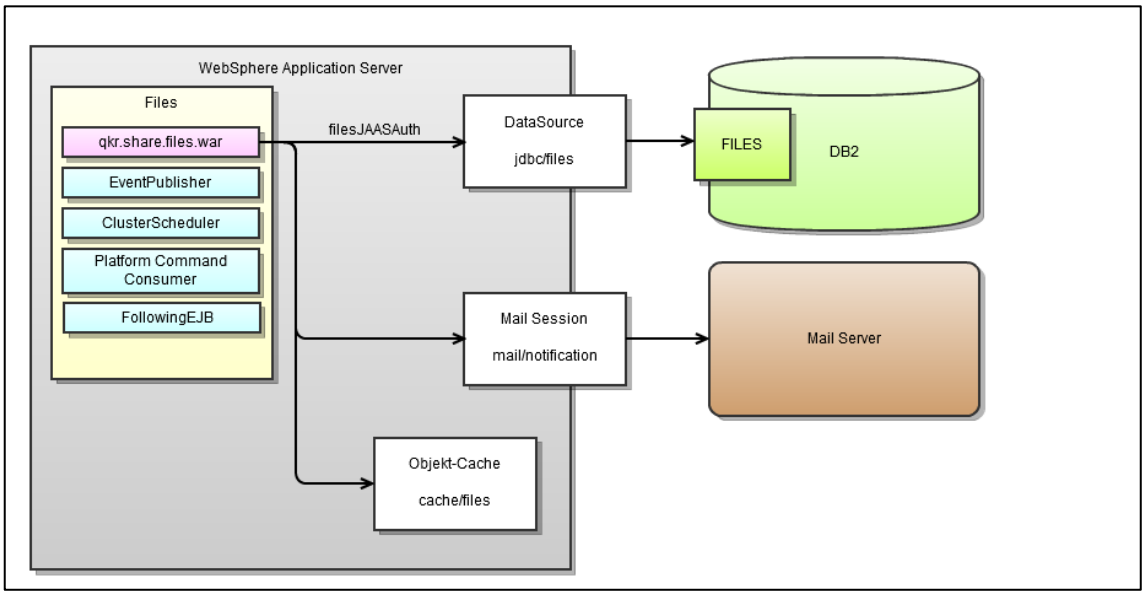


Abbildung 9 Aufbau der Files.ear

2.3.1.6 Profile

Die Anwendung „Profile“ dient einerseits als digitales Abbild eines jeden Mitarbeiters. Dadurch ist es möglich viele Informationen über Standort, Interessen, Kenntnisse und Kontaktinformation anzugeben, die dann von anderen Mitarbeitern durchsucht werden können, um einen geeigneten Ansprechpartner für ein Problem zu finden. [23]

Die Profile.ear liefert das Profil-System für IBM Connections. Dafür wird eine Datenbank- und Mail-Server-Anbindung benötigt.

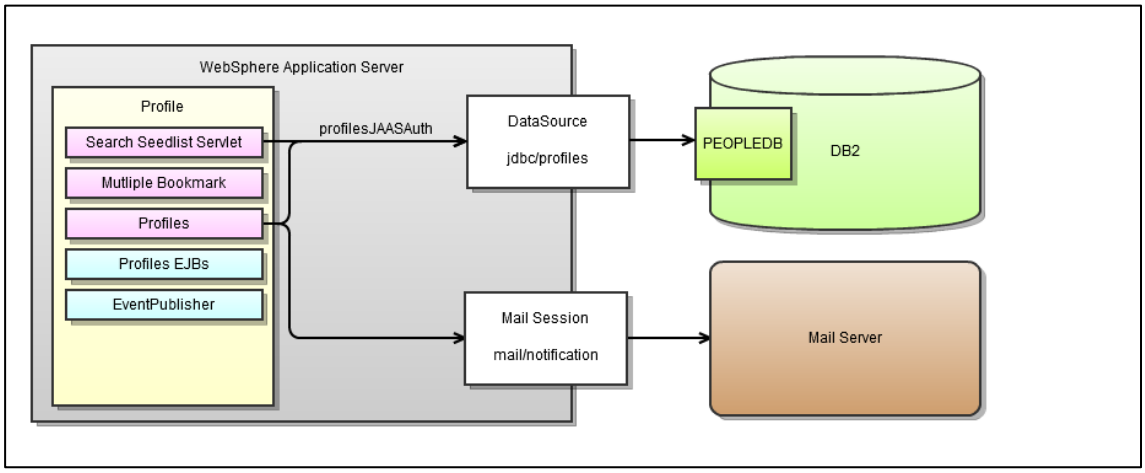


Abbildung 10 Aufbau der Profiles.ear

2.3.2 Connections als Gesamtpaket

IBM Connections bietet als Gesamtpaket die Vorteile, die die einzelnen Komponenten nicht bieten können. Durch die Kombination der einzelnen Anwendungen innerhalb von Communitys oder losgelöst von davon, ist es möglich für jeden Inhalt den passenden Container zu finden und dadurch ein zentrales Wissensnetzwerk aufzubauen. Durch die Freigabe und Veröffentlichung von Inhalten kann jeder Nutzer selbst bestimmen, welche Inhalte relevant sind und welche Inhalte am besten zu einem Thema passen. Durch die flexible Gestaltung kann IBM Connections als zentrale Wissensdatenbank oder zur Projekt- und Teamverwaltung genutzt werden. Ebenso ist es möglich mehrere Aspekte gleichzeitig zu verwenden, um dadurch eine Anlaufstelle für Information innerhalb des gesamten Unternehmens zu bieten, egal an welchem Standort man sich befindet. Durch die Kombination mit anderen Softwarelösungen wie Lotus Notes ist es möglich auf die Informationen innerhalb des Connections-System zuzugreifen und zu verwenden und so eine leichtere Bearbeitung der täglichen Aufgaben zu ermöglichen.

Um dies zu ermöglichen sind einige EAs nötig, die allein keinen Vorteil bringen. Eine dieser Anwendungen ist Homepage. Die „Homepage.ear“ ist eine der kleinsten Anwendungen in IBM

Connections. Sie ist die einzige Anwendung, die keine Anbindung an den Mail-Server besitzt, da sie lediglich als Anzeige-Plattform dient. Anders als vielleicht erwartet liefert die EA keine Widgets, obwohl ein Großteil der Web-Oberfläche daraus besteht. Diese Widgets werden zusammen mit den anderen Pflichtanwendungen „News“ und „Search“ integriert.

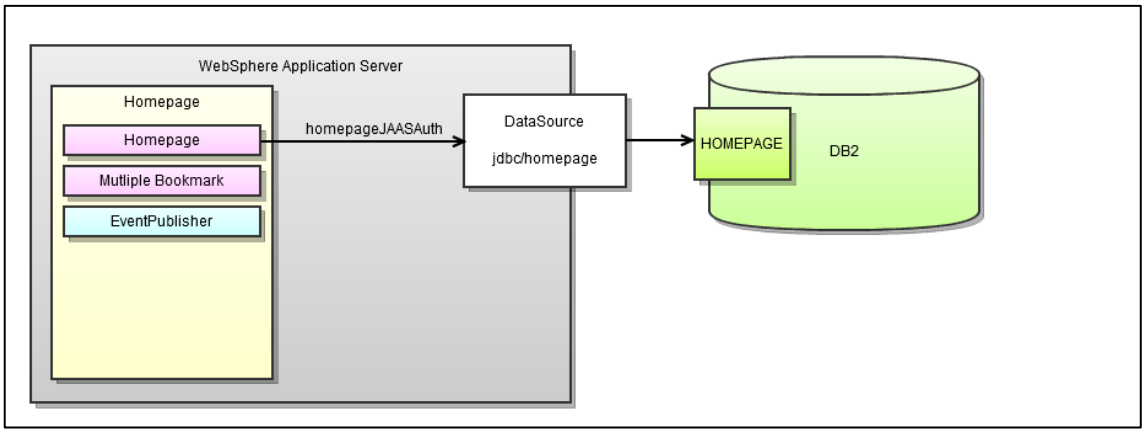


Abbildung 11 Aufbau der Homepage.ear

Die EA „News“ liefert Statusaktualisierungen. Sie beinhaltet diese Anwendung die Widgets für die Anwendung „Homepage“ und nutzt daher auch die gleiche Datenbank wie „Homepage“. Zusätzlich ist „News“ für die Sendung der Newsletter zuständig und nutzt dafür eine eigene Mail-Server-Anbindung.

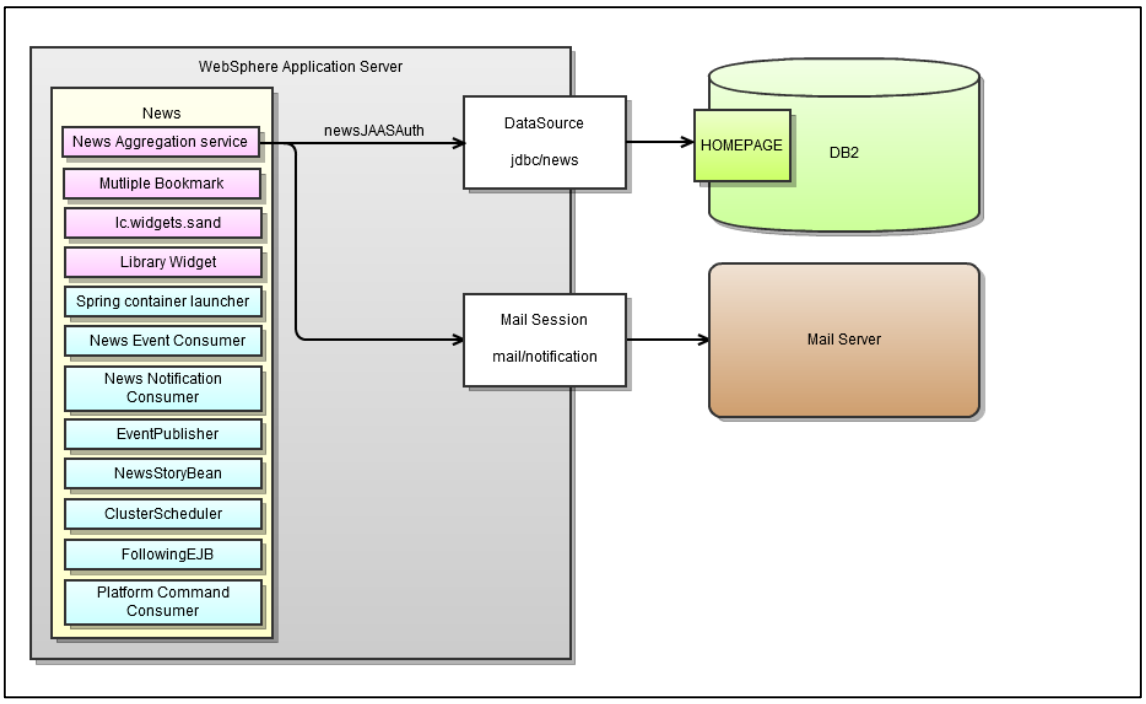


Abbildung 12 Aufbau der News.ear

Die EA „Search“ liefert ein Index- und Such-System („search.indexer“). „Search“ nutzt ebenfalls die Datenbank von „Homepage“.

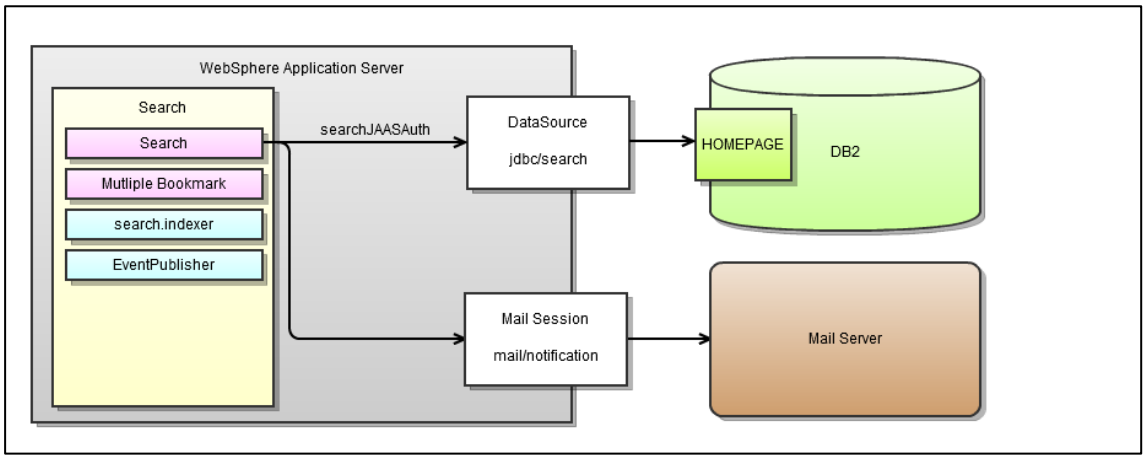


Abbildung 13 Aufbau der Search.ear

3 Analyse

In der Analyse soll untersucht werden an welchen Stellen IBM Connections Überblickseiten verwendet. Zusätzlich wird analysiert, was auf diesen Überblickseiten angezeigt werden kann und was nicht. Anschließend wird untersucht welche Schnittstellen IBM vorgegeben hat um Daten aus dem System auszulesen, aufzubereiten und anschließend wieder ins System zu integrieren.

3.1 Untersuchung des Ist-Zustandes und des Defizits

3.1.1 Überblickseiten

IBM Connections bietet in der Regel zwei Überblickseiten, die je nach Verwendungszweck unterschiedlich verwendet werden können. Zum einen gibt es die Homepage, die als Startseite der Anwendungen gesehen werden kann. Diese ist für jeden einzelnen individualisierbar und hat somit den Charakter einer Nachrichtenseite, in der zum Beispiel die letzten Änderungen angezeigt werden. Auf Abbildung 14 sind sieben einzelne Widgets dargestellt, die unterschiedliche Objekttypen anzeigen.

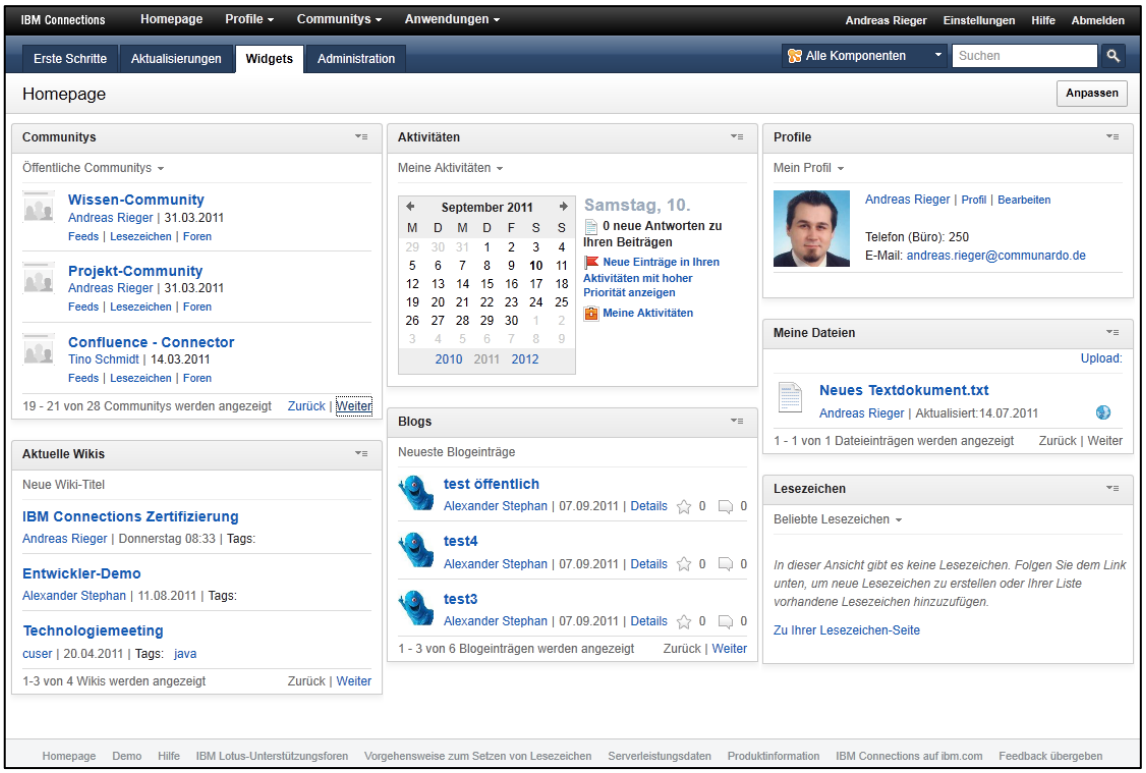


Abbildung 14 Beispiel-Homepage

Zum anderen bietet jede Community eine Überblickseite, in der die letzten Einträge und Änderung der jeweiligen Community angezeigt werden. Die Art und Anzahl der angezeigten

Einträge bezieht sich dabei auf die in der Community verwendeten Anwendungen. Abbildung 15 zeigt, wie eine Überblicksseite einer Community mit Blogs, Lesezeichen, Dateien, Aktivitäten und Foren aussieht.

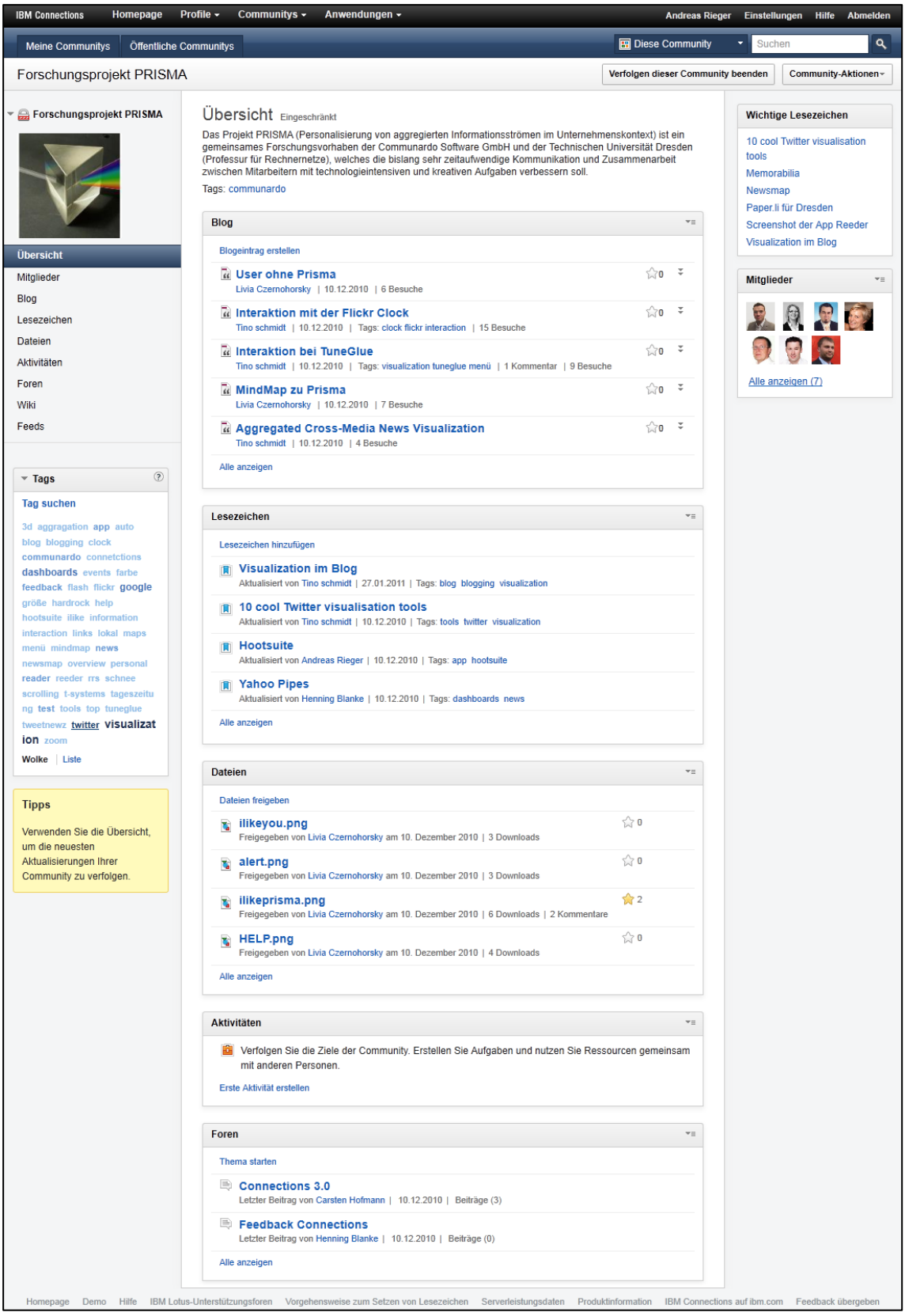


Abbildung 15 Beispiel-Überblicksseite einer Community

3.1.2 Homepage-Elemente

Auf der Homepage ist eine Vielzahl von Widgets bereits vordefiniert. Darunter werden fast alle Anwendungen abgedeckt. Es gibt Widgets für Aktivitäten, Blogs, Communitys, Dateien, Lesezeichen, Profile und Netzwerk sowie Wikis (siehe Tabelle 2). Teilweise gibt es Multiwidgets, die alle Funktionen der Widgets der gleichen Kategorie in einem Widget umsetzen. Dazu gibt es einen Schalter zum Auswählen des Modus.

Tabelle 2 Standard Homepage-Widgets [24]

Kategorie	Widgets
Aktivitäten	Meine Aktivitäten, Öffentliche Aktivitäten, Aktivitäten-Multiwidget
Blogs	Neuste Blogeinträge
Communitys	Meine Communitys, Öffentliche Communitys, Communitys-Multiwidget
Dateien	Meine Dateien, Mit mir gemeinsam genutzte Dateien
Lesezeichen	Beliebte Lesezeichen, Meine Lesezeichen, Meine Überwachungsliste, Neue Lesezeichen, Lesezeichen-Multiwidget
Profile	Mein Netz, Mein Profil, Profile-Multiwidget
Wikis	Aktuelle Wikis, Beliebte Wikis, Meine Wikis

Das Problem der Homepage-Elemente ist es, das nur die letzten Aktualisierungen der jeweiligen Einträge angezeigt werden. Man hat keine Möglichkeit den Inhalt zu filtern, um dadurch einen schnelleren Überblick zu erhalten. Man hat zum Beispiel keine Möglichkeit, die Einträge einer bestimmten Person anzuzeigen. Weiterhin existieren keine Widgets für Foren.

3.1.3 Community-Elemente

In einer Community-Überblicksseite gibt es die gleichen Probleme, wie auch in den Homepage-Elementen. Der Inhalt lässt sich nicht filtern. Zusätzlich werden die Inhalte der SubCommunitys nicht angezeigt. Ansonsten bietet die Community für jede Anwendung ein Widget [25].

3.2 Untersuchung der Programmschnittstellen

IBM Connections bietet Schnittstellen zur Erweiterung der Standardfunktionalität, die unabhängig von den verwendeten Komponenten genutzt werden kann. Einerseits liefert IBM ein „Service Provider Interface“ (SPI), anderseits gibt es eine „Application Programming Interface“ (API) mit dem auf die einzelnen Komponenten unter IBM Connections zugegriffen werden kann. Zusätzlich wird seit der IBM Connections Version 2.0 die IBM-spezifische iWidget-Spezifikation der Version 1.0 unterstützt. [26]

3.2.1 SPI

Das SPI besteht aus zwei Teilen. Das Event-SPI wird zur Ereignissteuerung verwendet. Dadurch könne Ereignisse abgefangen werden und eine besondere Ereignisbehandlung gestartet werden. Dies kann vor dem Ereignis (als Pre-Event-Handler) oder nach dem Event (als Post-Event-Handler) umgesetzt werden. Beide Ereignistypen bieten die Möglichkeit einer synchronen Abarbeitung, die jedoch bei falscher Anwendung zu Beeinträchtigungen des Systems führen kann, da die Ereignisbehandlung, den weiteren Programmablauf blockiert. Für die Eventbehandlung muss ein Event-Handler geschrieben werden, der das Package „SPI.jar“ verwendet. Diese „SPI.jar“ bringt jede Anwendung in INM Connections mit. In diesem Package gibt es abstrakte Klassen für Pre- und Post-Event-Handler. Nach der Implementierung muss der neu entstandene Programmteil in „external Library“-Verzeichnis des WebSphere Application Servers gepackt werden. Um die Event-Ausführung zu starten, müssen in der Ereignis-Konfiguration von IBM Connections („events-config.xml“) die neuen Event-Handler eingetragen werden. Dabei muss festgelegt werden auf welches Ereignis der Handler reagieren soll, wann die Abarbeitung erfolgt (Vor- oder Nachbearbeitung) und ob die Bearbeitung synchron oder asynchron erfolgen soll. [27]

Zusätzlich bietet das SPI eine Schnittstelle zur Integration einer eigenen Suchmaschine. Dies ermöglicht in einem Unternehmen zum Beispiel die Implementierung der unternehmensweiten Suche auch über Systemgrenzen hinaus und mit anderen Anwendungen. IBM verwendet dabei den Standard der Seed-List in der Version 1.0. Damit können Suchmaschinen die diesen Standard unterstützen angebunden werden. Dazu zählt IBMs OmniFind und auch Microsofts FAST Enterprise Search. [28]

3.2.2 API

Das API dient zum Zugriff auf die einzelnen Anwendungen. Es lässt sich damit Lese- und Schreibzugriffe realisieren. Das API setzt dabei auf das „Atom Publishing Protocol“, kurz AtomPub. [29]

Dies ermöglicht eine unabhängige Bearbeitung in verschieden Anwendungen, jedoch nur auf die Ressourcen, die IBM im AtomPub-Server dafür vorsieht. IBM setzt dabei auf einen Apache Abdera AtomPub Server der Version 0.4.0 vom 11. April 2008. Nachteil dieser Version ist, dass es die letzte als „Incubation“ geführt Version der Apache Software Foundation ist. Damit lag die Version zur internen Prüfung für die Aufnahme in die Stiftung. Diese wurde im November des Jahres aufgenommen. Anschließend wurden große Teile der Server- und Client-Software angepasst, sodass die neuen Versionen nach 1.0, die im Jahre 2010 und später veröffentlicht wurden, teilweise nicht kompatibel sind. Dies hat zur Folge, dass die Neuerungen und Verbesserungen, die in den höheren Versionen eingeführt wurden, nicht verwendet werden können. [30]

Jede Anwendung von IBM Connections verfügt über einen Abdera AtomPub-Server. Die Zugriffsmöglichkeiten und die dazu vorhandene Dokumentation sind jedoch sehr unterschieden. Es wurde festgestellt, dass die Zugriffsdokumentationen der neusten Anwendungen nicht oder nur sehr grob vorhanden sind. IBM nutzt dies um die neue Programmkomponente zu überprüfen und bis zum nächsten Release zu überarbeiten. Mit dem Release wir dann meistens auch die dazugehörige Dokumentation veröffentlicht. IBM empfiehlt auch nur die dokumentierte Funktionalität der Connections API zu verwenden, da alles was nicht dokumentiert ist, beim nächsten Release oder Update abgeändert sein kann.

Da für die geplante Entwicklung nur lesender Zugriff benötigt wird, soll dies hier im Vordergrund der Untersuchung liegen. Die Auflistung der APIs bezieht sich nur auf die für die spätere Programmierung relevanten Anwendungen. IBM Connections bietet noch weitere APIs, die zum Beispiel eine Suchabfrage ermöglichen.

3.2.2.1 Activities API

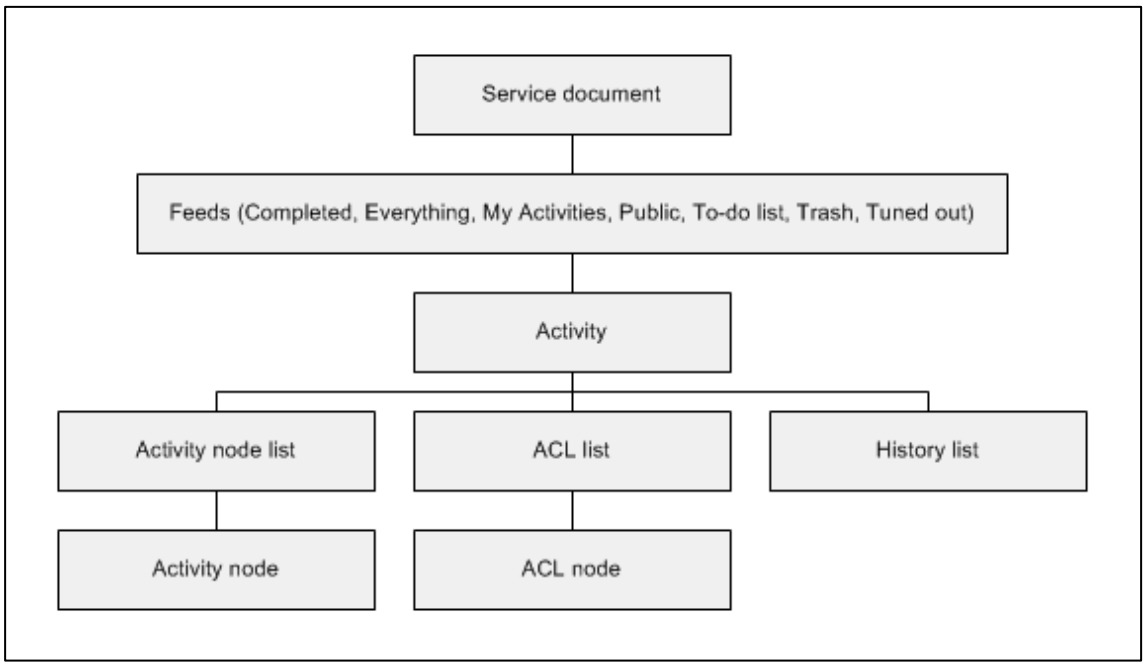


Abbildung 16 Zugriffshierachie bei Aktivitäten [31]

Das API von Activitites bietet Zugriff auf alle Daten, die in Activities hinterlegt sind. In der ersten Ebene können Aktivitäten abgefragt werden. In den ausgelieferten Feed-Entries können viele Aktivitäten-Eigenschaften herausgenommen werden. Mithilfe der *ID des Eintrags kann über die URL „…/activities/service/atom2/descendants?nodeUuid=*ID“ direkt auf die Aktivität zugegriffen und die Kindelemente auslesen werden. Die Kindelemente können Abschnitte (Sektion), Einträge (Entry) oder Aufgaben (Todo) sein. In dieser Reihenfolge können sie auch in einer Baumstruktur angelegt sein.

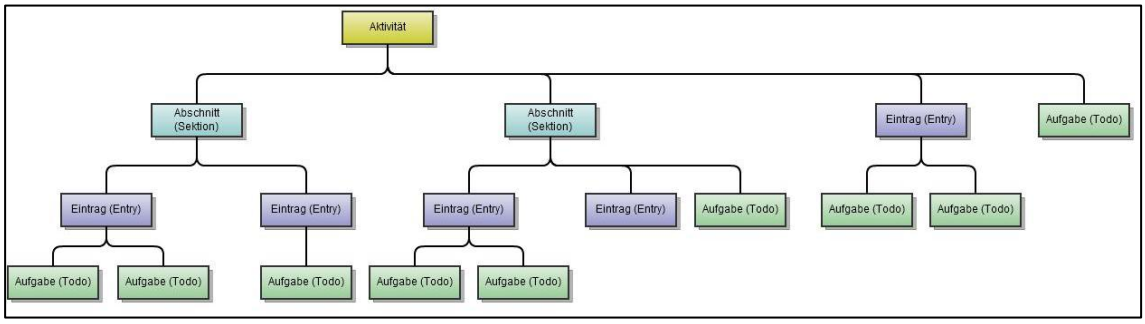


Abbildung 17 Mögliche Baumstruktur bei Aktivitäten

3.2.2.2 Blogs API

Das Blogs API bietet ähnlich wie das Activities API Zugriff auf alle Inhalte der Blogs-Anwendung. Für die Abfragen gibt es jedoch keine feste URL. Die URL muss immer über die Blogs-Homepage geleitet werde, dessen Zugriffspunkt wiederum variable ist. In jeder Connections-Umgebung kann die Homepage unterschiedlich definiert werden. Der Zugriff auf die Blogs-Daten erfolgt über „…/blogs/roller-ui/rendering/feed/<Blogs-Homepage>/entries/atom“ [32].

Über die Server-Schnittstelle der Blogs-Anwendung gibt es keinen Zugriff auf „Meine Blogs“. Es werden lediglich „Öffentliche Blogs“ und die letzten Aktualisierungen angeboten.

3.2.2.3 Bookmarks API

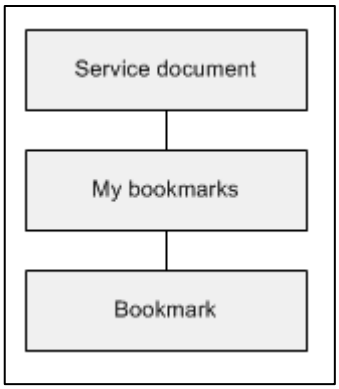


Abbildung 18 Zugriffshierarchie bei Bookmarks [33]

Das Bookmarks API ist die Schnittstelle um Informationen zu Lesezeichen zu erhalten. Der Zentrale Zugriffspunkt für Lesezeichen ist „…/dogear/atom“ . Durch Parameter kann die ursprünglich öffentliche Ansicht auf die private beschränkt werden. Parameter dafür sind „email“ oder „userid“. Wenn ein solcher Parameter vorhanden ist, kann man die Lesezeichen der übergebenen Person betrachten.

3.2.2.4 Communities API

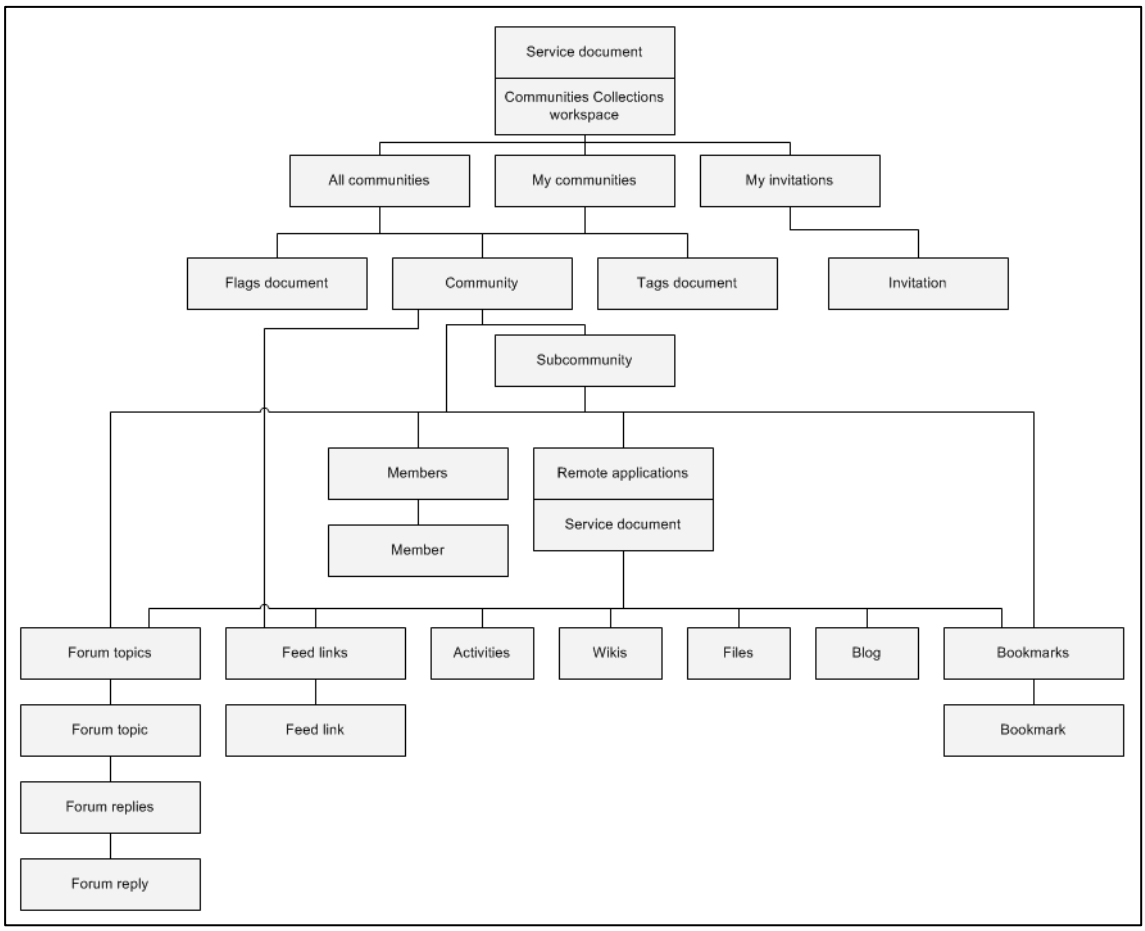


Abbildung 19 Zugriffshierarchie bei Communitys [34]

Das Communities API bietet Zugriff auf alle Ressourcen der Communitys. Es können spezielle Feeds für „Öffentliche“ oder „Meine Communitys“ abgefragt werden. „Meine Communitys“ sind zusätzlich noch nach der Rolle innerhalb der Community unterscheidbar. Verfügbar sind Mitglied, Eigentümer oder Follower. Zugriff existiert über „…/communities/service/atom/communities/<all oder my>“ [35]. Über die Community-Einträge im Feed ist ein Zugriff auf Community-Anwendungen möglich, sodass diese ausgelesen werden können. Ein Zugriff auf die Mediengallery wird nicht unterstützt.

3.2.2.5 Files API

Den Zugriff auf die persönliche Dateibibliothek bietet das Files API. Diese lässt sich in „Öffentlich“, „Meine“, „Von mir geteilte“, „Für mich freigegebene“ und „Angehängte Dateien“ unterscheiden. Der Zugriff entsteht über „/files/basic/api/userlibrary/<Userid oder „anonymous“ für öffentlich>/feed“ [36]. Über diesen Feed lassen sich alle Dateien der Bibliothek abfragen. In den einzelnen Einträgen sind sämtliche Informationen der Datei enthalten, darunter auch die Version. Zusätzlich verfügt jeder Eintrag eine URL zum Download der Datei.

3.2.2.6 Forums API

Das Forums API dient zur Navigation innerhalb von Foren. Es bietet dabei Zugriff auf öffentliche Foren, meine Foren, meine Themen und meine offene bzw. beantworteten Fragen. Die Komponenten sind wiederum nach der Rolle des Nutzers unterteilt, wobei Mitglied, Eigentümer oder Follower zur Auswahl stehen. Mithilfe des APIs lassen sich die Schlagwort-Informationen zu den öffentlichen und eigenen Foren auslesen. Für die Schlagworte von Themen und Fragen, muss jedes Thema bzw. Frage erneut abgefragt werden. [37]

3.2.2.7 Wikis API

Das Wikis API ermöglicht den Zugriff auf die angelegten Wikis. Es bietet in der ersten Ebene keinen Zugriff auf die enthaltenen Wiki-Seiten, dazu wäre eine weitere Abfrage je Wiki nötig.

3.2.3 Vergleich von API und SPI

Tabelle 3 zeigt die Unterschiede von API und SPI im direkten Überblick.

Tabelle 3 Vergleich API und SPI

	API	SPI
Grundlage	basiert auf XML over HTTP (Atom)	basiert auf Java
Funktionsumfang	Zugriff auf von IBM zur Verfügung gestellte Funktionen	Eventbehandlung und Erweiterung der Suche
Nutzung	Standardschnittstelle zum Datenauslesen und -manipulation	Standardschnittstelle zur Eventbearbeitung
Kompatibilität zukünftige Version	kompatibel für dokumentierte Funktionen	keine garantierte Kompatibilität

3.2.4 Datenbank-Analyse und Datenbank-Zugriff

Fast jede Anwendung in Connections verwendet seine eigene Datenbank. Die Datenbanken verwenden jeweils eigene IDs für teilweise die gleichen Objekte, sodass beispielsweise ein Nutzer in den Datenbeständen von Foren nicht die gleiche ID, wie in Profilen besitzt. Eine Direkte Abfrage ist somit nur über Umwege möglich.

IBM sieht keine direkten Datenbank-Zugriffe von Erweiterungen vor.

3.2.5 iWidgets

Communitys, Homepage und Profile unterstützen die Erweiterung durch iWidgets. IBM liefert aktuell die iWidget-Spezifikation 1.2 aus, IBM Connections unterstützt derzeit die Version 1.0.

iWidgets sind XML-Dateien, die externe Inhalte in die bestehende Funktionalität anzeigen. Dies kann statisch innerhalb der XML-Datei erfolgen oder aber dynamische mit JavaScript geschehen. Im ersten Fall wird ein HTML-Stück innerhalb der XML-Datei hinterlegt, welches dann angezeigt wird. Dies kann auch ein iFrame sein. Im zweiten Fall wird eine JavaScript-Ressource hinterlegt, die dann bei definierten Ereignissen, wie „onLoad“, „onView“oder „onEdit“, geladen wird. Damit ist es möglich HTML-Teile mittels AJAX aus einer anderen Web-Anwendung zu laden, wenn das iWidget angezeigt wird, um dem Nutzer so dynamische Inhalte anzuzeigen. [38]

IWidgets für Homepage lassen sich vom Administrator auf der Administrationsseite der Homepage-Anwendung hinzufügen und freischalten. Für Communitys und Profile muss die widget-config.xml im Konfigurationsordner des WebSphere Application Servers angepasst werden. Im Anschluss muss der Server neu gestartet werden.

3.2.6 Zugriff auf globale Funktionen

Einbindung der vCard

Durch Einbindung eines HTML-Ausschnittes wird die IBM Connections vCard auch in externen Anwendungen angezeigt [39]. Dieser Ausschnitt besteht aus einem JavaScript-Teil, der eine externe JavaScript-Ressource lädt.

Code-Ausschnitt 1 vCard-Einbindung Teil 1

```
<script type="text/javascript"
     src=".../profiles/ibm_semanticTagServlet/javascript/semanticTagService.js"></script>
```

Zusätzlich wird noch ein Ausschnitt eingebunden, der den Container für die vCard darstellt.

Code-Ausschnitt 2 vCard-Einbindung Teil 2

```
<div class="vcard X-person-display-inline">
     <span class="fn" style="display:none;">
          <Benutzername>
     </span>
     <span class="email" style="display:none;">
          <Benutzer-E-Mail-Adresse>
     </span>
</div>
```

3.2.7 Fazit der Programmschnittstellen-Analyse

Für die Erweiterungen werden iWidgets für Homepage und Communitys angelegt. Diese zeigen bestehende Inhalte von IBM Connections an, die mithilfe des APIs abgefragt und anschließend zur Anzeige aufbereitet werden. Dazu zählt eine Filterung der Inhalte durch verschiedene Kriterien, wie Autoren, Titelbeschreibung oder Schlagwort-Belegung.

Für die Erweiterungen wird eine Kombination aus JavaServerPages und Servlets verwendet um eine MVC-Anwendung zu realisieren. Die Servlets dienen dabei der Beschaffung und Aufbereitung der Daten, während die JavaServerPages hauptsächlich für die Darstellung zuständig sind. Das SPI wird nicht für die Erweiterungen verwendet, da es nicht für die Art von Erweiterungen vorgesehen ist, die geplant sind.

Es soll auf einen direkten Zugriff auf die Datenbanken verzichtet werden, um die Update-Fähigkeit zu gewährleisten. Zusätzlich soll durch die Erweiterung keine Sicherheitsschwachstelle entstehen.

4 Realisierung

In der Analyse wurden die Schnittstellen vorgestellt, die für eine Erweiterung von Connections in Betracht gezogen werden können. Die Möglichkeiten die IBM bietet halten sich in Grenzen, sodass es keine Wahl über die eingesetzten Programmiersprachen und Technologien zulässt. Die Realisierung werden Widgets sein, die die iWidget-Spezifikation von IBM einhalten. Dabei wird jedoch kein statischer Text angezeigt, sondern mittels JavaScript dynamische Inhalte abgerufen die wiederum einer Enterprise-Application in Java generiert werden. Diese EA beinhaltet Funktionen zum Laden der verfügbaren Daten mit Hilfe Connections API, das Filtern der Daten und die Darstellung mittels JavaServerPages.

4.1 Überblick

Um eine spätere Verwendung und eine leichtere Anpassung zu ermöglichen, wurde entschieden, sich strikt an das Model-View-Controller-Muster zu halten. Dieses sorgt für die Trennung der Programmbereiche für Datenmodell, Präsentation und Programmsteuerung (siehe Abbildung 20; der Prozess beginnt auf der Homepage im externen System).

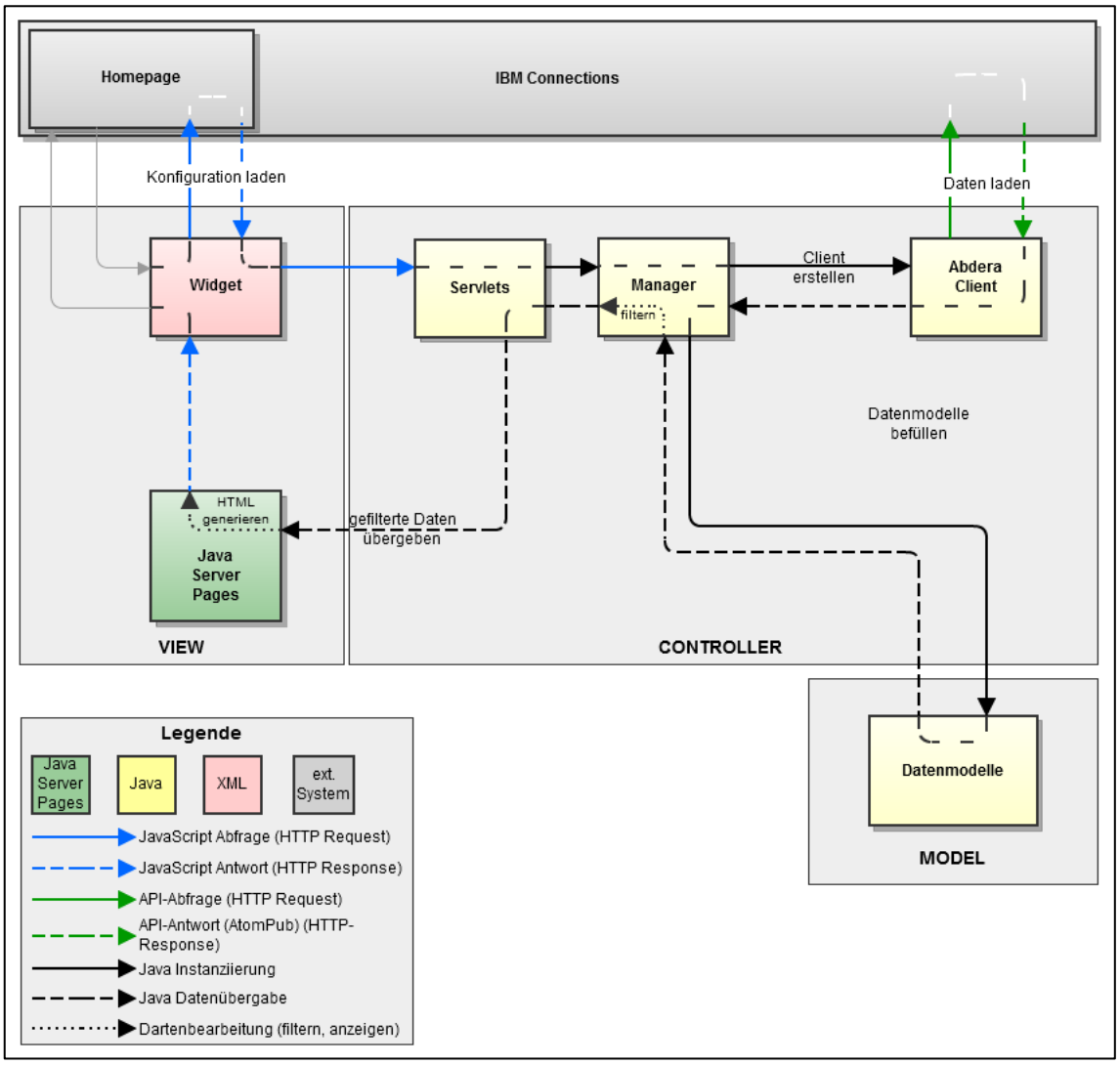


Abbildung 20 Model-View-Controller

4.1.1 Model

Für jedes Datenelement in Connections wird ein eigener Datentyp erstellt, der die auszulesenden Eigenschaften beinhaltet. Dazu zählen Objekte für Aktivitäten, Lesezeichen, Communitys, Foren und Wikis. Diese befinden sich im Komponenten-Paket. Zusätzlich gibt es noch übergreifende Objekte für Taglisten und Personen.

4.1.2 Controller

Zu dem Controller Paket gehört ein Abdera-Client, der die Abfrage in das Connections-System realisiert. Ebenfalls dazu gehören die Servlets, die als Eingangstor für die anzuzeigenden Webseiten gelten.

Im Manager-Paket sind die Programm-Komponenten enthalten, die die Schnittstelle zwischen Abdera-Client und Datenmodelle bilden. Diese beinhalten die meiste Programm-Logik. Sie sind

auch für Filterung der Daten zuständig. Die Manager leiten die Daten an die grafische Ausgabe der Webanwendung.

Zur Programm-Steuerung gehören auch die JavaScript-Dateien, die für einen flüssigen Ablauf der Anwendung zuständig sind. Diese sind auch für das Nachladen der Daten zuständig, ohne die komplette Homepage neu zu laden.

4.1.3 View

Die Anzeige (der View) wird über JavaServerPages realisiert. Zusätzlich wurde mittels CSS und Grafiken die Darstellung der Widgets an das Aussehen der bereits bestehenden Komponenten angelehnt, um eine größtmögliche optische Verschmelzung zu erreichen.

4.2 Komponenten

4.2.1 iWidget

Das Widget ist eine einfache XML-Datei. Diese muss der iWidget-Spezifikation von IBM entsprechen. Im Kopf der XML-Datei wird der Namespace für das iWidget festgelegt. Im Anschluss werden die unterstützten Modi (supportedModes) angegeben, in diesem Fall „view“ und „edit“. Zusätzlich wird noch der Standard-Modus angegeben, in dem das Widget gestartet wird, wenn die Homepage geladen wird (mode=„view“). Im iScope wird der Geltungsbereich der Variablen festgelegt. (siehe Code-Ausschnitt 3)

Im Anschluss werden die Variablen festgelegt, die gespeichert werden können. Alle Variablen außer „count“ und „app“ dienen der Filterung der anzuzeigenden Daten und dürfen unbenutzt sein. Lediglich „count“ besitzt als Standardwert „3“. Dies beschreibt die maximale Anzahl der Elemente, die je Seite angezeigt werden können. Alle Daten werden in einem ItemSet abgelegt, welches bequem per JavaScript auslesbar und beschreibbar ist. (siehe Code-Ausschnitt 3; Kommentar „Widget Attributes“)

Nachfolgend werden die einzubindenden JavaScript-Ressourcen festgelegt. In diesem Fall ist es eine spezielle Ressource mit der ID „EAWHID“. In dieser Datei sind alle JavaScript-Funktionen hinterlegt, die von der Webanwendung verwendet wird (siehe 4.2.2.3 EAWH.js). Zusätzlich wird noch das JavaScript-Framework JQuery (ID:JQuery) eingebunden und ein JQuery-Plugin (ID:JQueryUI). Auch das CSS wird als Ressource eingebunden (ID: STYLEID). (siehe Code-Ausschnitt 3; Kommentar „JavaScript Resource“ und „CSS Resource“)

Letztendlich wird ein Content bereitgestellt, der, je nachdem welcher Modus gewählt wurde, geladen wird. In beiden Modi wird nur ein „div“-Element mit einer ID geladen, welches als

Einhängepunkt für JavaScript dient. (siehe Code-Ausschnitt 3; Kommentar „VIEW Mode“ und „EDIT Mode“)

Code-Ausschnitt 3 Widget XML

```
<?xml version="1.0" encoding="UTF-8" ?>
<iw:iwidget id="EAWH" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:iw="http://www.ibm.com/xmlns/prod/iWidget" supportedModes="view edit"
    mode="view" iScope="EAWHScope">

    <!-- Widget Attributes-->
    <iw:itemSet id="attributes">
        <iw:item id="app" readOnly="false"/>
        <iw:item id="typeopt" readOnly="false"/>
        <iw:item id="titleopt" readOnly="false"/>
        <iw:item id="title" readOnly="false"/>
        <iw:item id="author" readOnly="false"/>
        <iw:item id="contributor" readOnly="false"/>
        <iw:item id="tagopt" readOnly="false"/>
        <iw:item id="tags" readOnly="false"/>
        <iw:item id="communityopt" readOnly="false"/>
        <iw:item id="community" readOnly="false"/>
        <iw:item id="communityId" readOnly="false"/>
        <iw:item id="count" readOnly="false" value="3"/>
    </iw:itemSet>

    <!-- JavaScript Resource-->
    <iw:resource id="EAWHID" src="../javascript/EAWH.js" />
    <iw:resource id="JQuery" src="../javascript/jquery-1.6.2.min.js" />
    <iw:resource id="JQueryUI" src="../javascript/jquery.autocomplete.js" />

    <!-- CSS Resource-->
    <iw:resource src="../stylesheet/style.css" id="STYLEID" />

    <!-- VIEW Mode-->
    <iw:content mode="view">
        <![CDATA[
            <div id="VIEW_DIV">Loading...</div>
        ]]>
    </iw:content>

    <!-- EDIT Mode-->
    <iw:content mode="edit">
    <![CDATA[
            <div id="CONFIG_DIV">Loading...</div>
        ]]>
    </iw:content>

</iw:iwidget>
```

4.2.2 JavaScript

Das verwendete JavaScript besteht genau genommen aus vier Teilen. Einerseits wird das von Connections eingesetzte Dojo genutzt. Dies wird für alle AJAX-Abfragen verwendet und dient im Allgemeinen als meist genutzte Bibliothek. Das JQuery-Framework wird benötigt um die AutoComplete-Funktion für Tags und Communitys zu ermöglichen. Diese wird durch das JQuery-Plugin „AutoComplete“ gelöst. Zusätzlich wird noch ein Großteil der Programmfunktionalität in einer speziellen JavaScript-Datei hinterlegt. Dies ist die „EAWH.js“.

4.2.2.1 Dojo

Dojo wird automatisch in Connections eingebunden und muss nicht erneut ausgeliefert werden. Daher ist eine Nutzung von vorn herein möglich und empfehlenswert. Es wird verwendet um einfache DOM-Manipulationen und AJAX-Requests durchzuführen, welche alle in der „EAWH.js“ beschrieben sind.

4.2.2.2 JQuery und AutoComplete

AutoComplete ist eine Funktion, die bei der Eingabe in ein Textfeld Vorschläge aus einer Liste von Wörtern automatisch vorschlägt. Dies ermöglicht die genaue Eingabe von Daten und verhindert damit das falsch schreiben.

Dojo bietet eine AutoComplete-Funktion, die jedoch nicht das gewünschte Ergebnis gezeigt hat. Es gibt keine Möglichkeit mehrere Einträge auszuwählen und nebeneinander anzuzeigen. Diese Funktionalität hätte einen erheblichen Mehraufwand bedeutet, sodass entschlossen wurde dafür das JQuery-Framework mit dem AutoComplete-Plugin zu verwenden.

4.2.2.3 EAWH.js

In der Datei EAWH.js sind alle JavaScript-Funktionen gebündelt, die innerhalb der Webanwendung verwendet werden. Im nachfolgenden sollen die Funktionen beschrieben werden die umgesetzt wurden.

onView()

Die onView-Funktion wird geladen, wenn sich das Widget im „View“-Modus befindet. Aufgabe der Funktion ist es, die Konfiguration zu laden (siehe Code-Ausschnitt 4) und die Variable „app“ auszuwerten. Sollte die Variable „app“ gesetzt sein, wird die Konfiguration an ein Servlet gesendet, welches dann die darzustellenden Daten bereitstellt. Diese werden dann an den Anhängepunkt in der Widget-Definition eingefügt. Sollte „app“ nicht gesetzt sein, wird die Funktion „changeToEdit“ aufgerufen und dadurch der Modus zu „Edit“ gewechselt.

Code-Ausschnitt 4 Laden der Variablen

```
this.iContext.getiWidgetAttributes().getItemValue("VARIBALE")
```

onEdit()

Die onEdit-Funktion wird geladen wenn sich das Widget im „Edit“-Modus befindet. Ziel der Funktion ist es die globale Konfigurationsseite (siehe Abbildung 21) zu laden, welche von einem Servlet bereitgestellt wird.

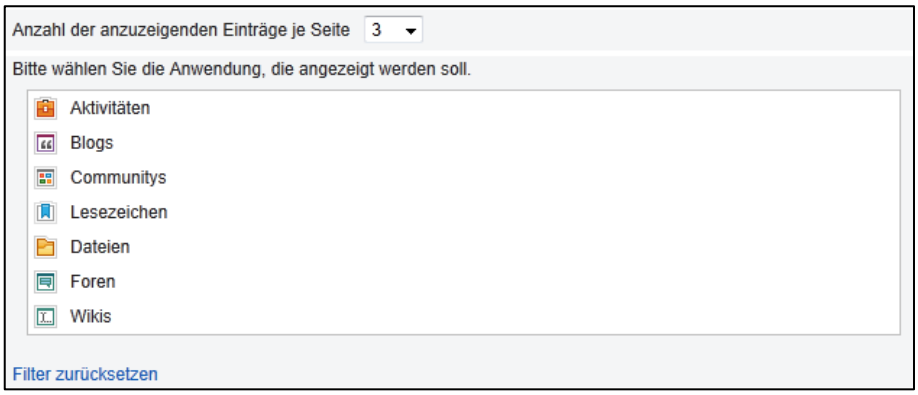


Abbildung 21 Globale Konfiguration

onLoading()

Die Funktion onLoading wird immer dann aufgerufen, wenn Inhalte nachgeladen werden müssen. Dies geschieht zu Beginn der onView Funktion, da im Anschluss der komplette Programmablauf ausgeführt werden muss. Abbildung 22 zeigt das Lade-Bild. Es entspricht dem Logo der Communardo Software GmbH in einer Rotation-Animation.

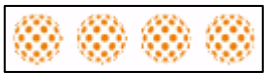


Abbildung 22 Lade-Bild (Einzelbilder separiert)

displaySub(element)

Die displaySub-Funktion sorgt dafür, dass die Unterelemente in der globale Konfigurationsseite (siehe Abbildung 23 und Abbildung 21) geladen werden. Dazu wird das Elternelement „element“ übergeben.

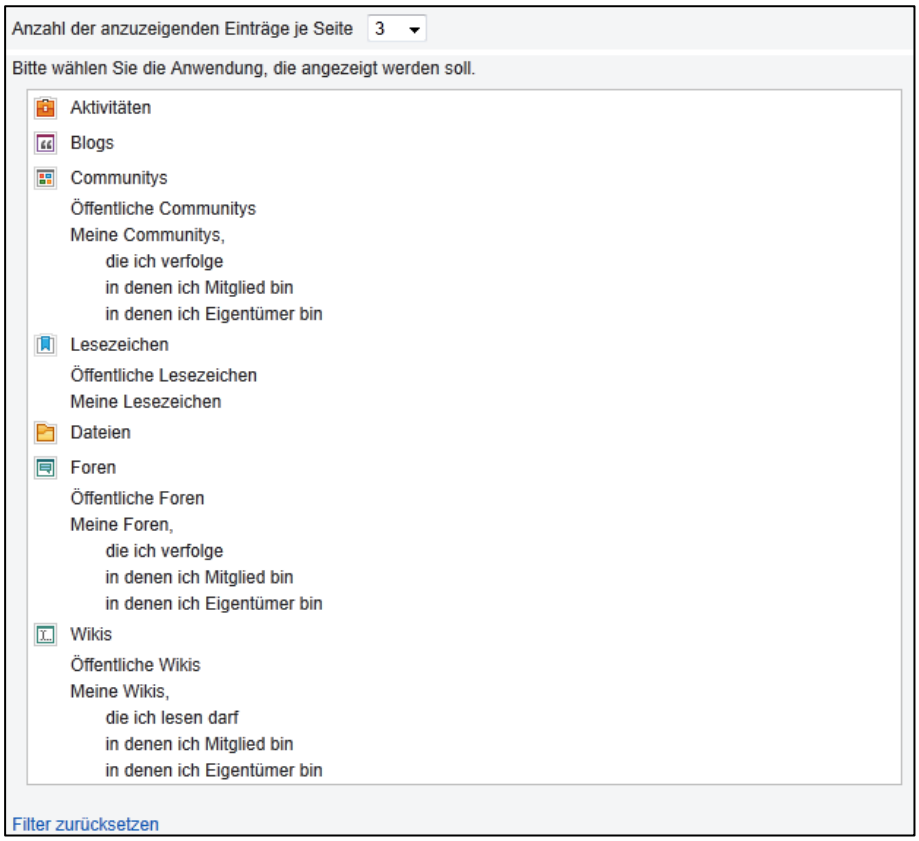


Abbildung 23 Globale Konfiguration mit angezeigten Unterelementen

nextPage(element)

Die Funktion nextPage sorgt dafür das von der globalen Konfigurationsseite die Element-Seite angezeigt wird, an der detaillierte Einstellungen vorgenommen werden können (siehe Abbildung 24 am Beispiel von öffentliche Foren).

Abbildung 24 Detail-Konfiguration von "öffentliche Foren"

display(option)

Die display-Funktion sorgt dafür, dass die Konfigurationselemente der Detailseite aufgeklappt werden können und somit eingestellt werden können. Abbildung 25 zeigt die Detailseite mit geöffneten Optionen für Autor und Community.

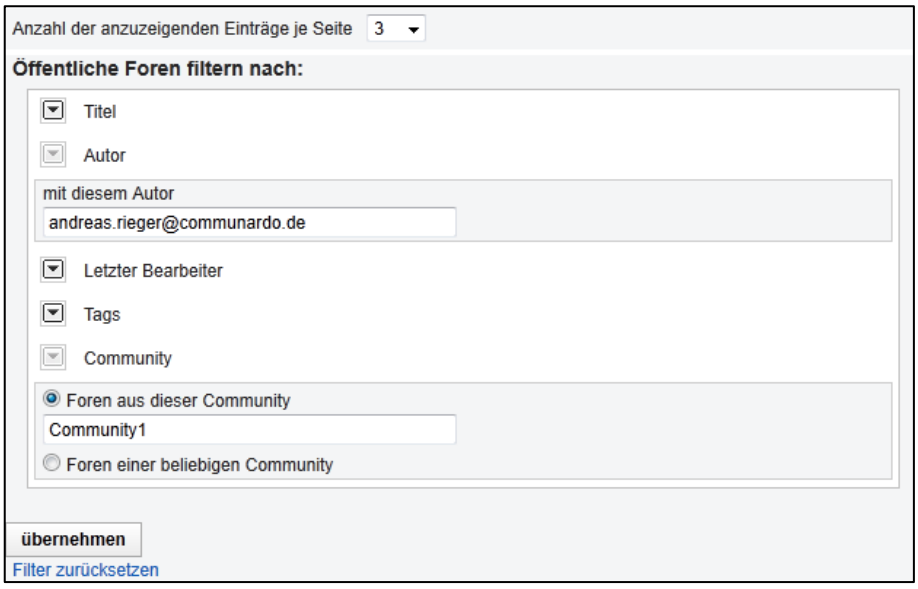


Abbildung 25 Detail-Konfiguration von "öffentliche Foren" mit teilweise geöffneten Optionen

loadCommunities()

Diese Funktion sorgt dafür, dass beim Laden einer Detailseite auch eine Liste der möglichen Communitys geladen wird. Die Liste wird für die AutoComplete-Funktion für das Community-Option-Feld benötigt (siehe Abbildung 26). Die AutoComplete-Funktion wurde hier mit einer „MustMatch“-Option angelegt. Dies bedeuted, dass die Eingabe nur erlaubt wird, wenn der getippte Buchstabe zu einer Community in der geladen Liste passt. Wenn sich zum Beispiel keine Community mit dem Anfangsbuchstaben „A“ in der Liste befindet, kann der Nutzer kein „A“ eingeben. Die Eingabe wird dann gelöscht.

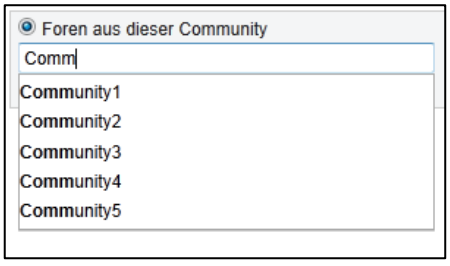


Abbildung 26 AutoComplete-Funktion für Communitys

loadTags()

Die loadTags-Funktion liefert im Effekt das gleiche wie die loadCommunities-Funktion mit dem Unterschied, dass die Schlagwörter anwendungsspezifisch geladen werden. Zusätzlich wird eine AutoComplete-Funktionalität für die Tag-Option realisiert, in der auch mehrere Schlagwörter durch Leerzeichen getrennt eingegeben werden können. Die Mehrfach-Eingabe wird in Abbildung 27 und Abbildung 28 gezeigt.

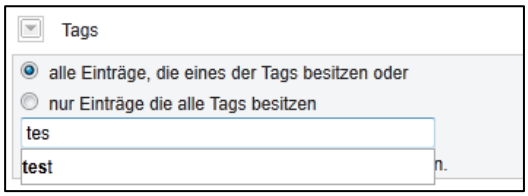


Abbildung 27 AutoComplete-Funktion für Tags; erste Eingabe

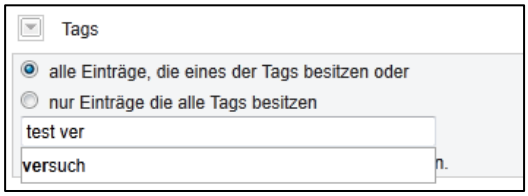


Abbildung 28 AutoComplete-Funktion für Tags; zweite Eingabe

saveConfig()

Nach dem Klicken auf „Übernehmen“ wird diese Funktion aufgerufen um die eingetragenen Daten zu speichern (siehe Code-Ausschnitt 5). Im Anschluss wird mit der Funktion „changeToView“ in den Anzeige-Modus (View) gewechselt.

Code-Ausschnitt 5 Speichern der Variablen

```
// für alle Variablen-Elemente ausführen
this.iContext.getiWidgetAttributes().setItemValue("VARIABLE", VARIABLENWERT);

// am Ende einmalig auszuführen
this.iContext.getiWidgetAttributes().save();
```

resetConfig()

Diese Funktion wird ausgeführt, wenn der Nutzer auf „Filter zurücksetzen“ klickt. Dies bewirkt, das die globale Konfigurationsseite neu geladen wird ohne die bisher eingegebenen Daten gespeichert werden.

pagination(start, end, page, direction)

Die pagination-Funktion realisiert das Blättern in den Ergebnissen (siehe Abbildung 29). Der Wert von „direction“ gibt dabei an, in welcher Richtung geblättert werden soll. Je nach dem wird dann der neue Startwert aus den anderen 3 Variablen berechnet und mit Hilfe der onView-Funktion angezeigt. „start“ ist dabei die Nummer des ersten angezeigten Elements, „end“ die des letzten Elements und „page“ die der Anzahl der maximal anzuzeigenden Elementen.

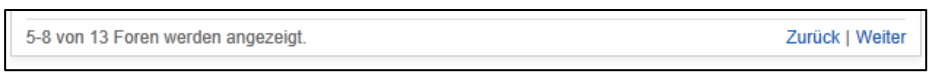


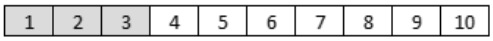
Abbildung 29 Seiten-Auswahl- Balken am unteren Ende des Widgets (Pagination)

Beispiel:

Es gibt 10 Elemente von denen die Elemente 4 bis 6 angezeigt werden. Die maximale Anzahl beträgt 3.

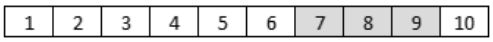
Wenn die pagination-Funktion nun mit den Werten 4 (erstes angezeigte Element) für `start`, 6 (letztes angezeigte Element) für `end` und 3 für `page`. Als `direction` wird „previous“ gewählt und damit soll eine Seite zurück geblättert werden.

Die neue Startwert errechnet sich aus `start` - `page` und lautet somit 1. Angezeigt wird die maximale Anzahl ab ersten Element.



Wenn die Funktion pagination wie zu Beginn mit den Werten 4,6 und 3 aufgerufen wird, jedoch mit der `direction` „next“ wird eine Seite nach weiter geblättert.

Der neue Startwert errechnet sich dann aus dem `end`-Wert +`1` und lautet damit 7. Angezeigt werden maximal 3 Elemente ab dem 7. Element.



Eine interne Prüfung in den Java-Funktionen verhindert, dass mehr Elemente geladen werden als vorhanden sind. Sollte also nach dem letzten Beispiel, die nächste Seite geladen werden, wird nur das 10. Element angezeigt.

4.2.3 Java

Ein Großteil des Programms wurde in Java realisiert. Als Grundlage dient dafür die JVM des WebSphere Application Servers.

4.2.3.1 Klassendiagramm

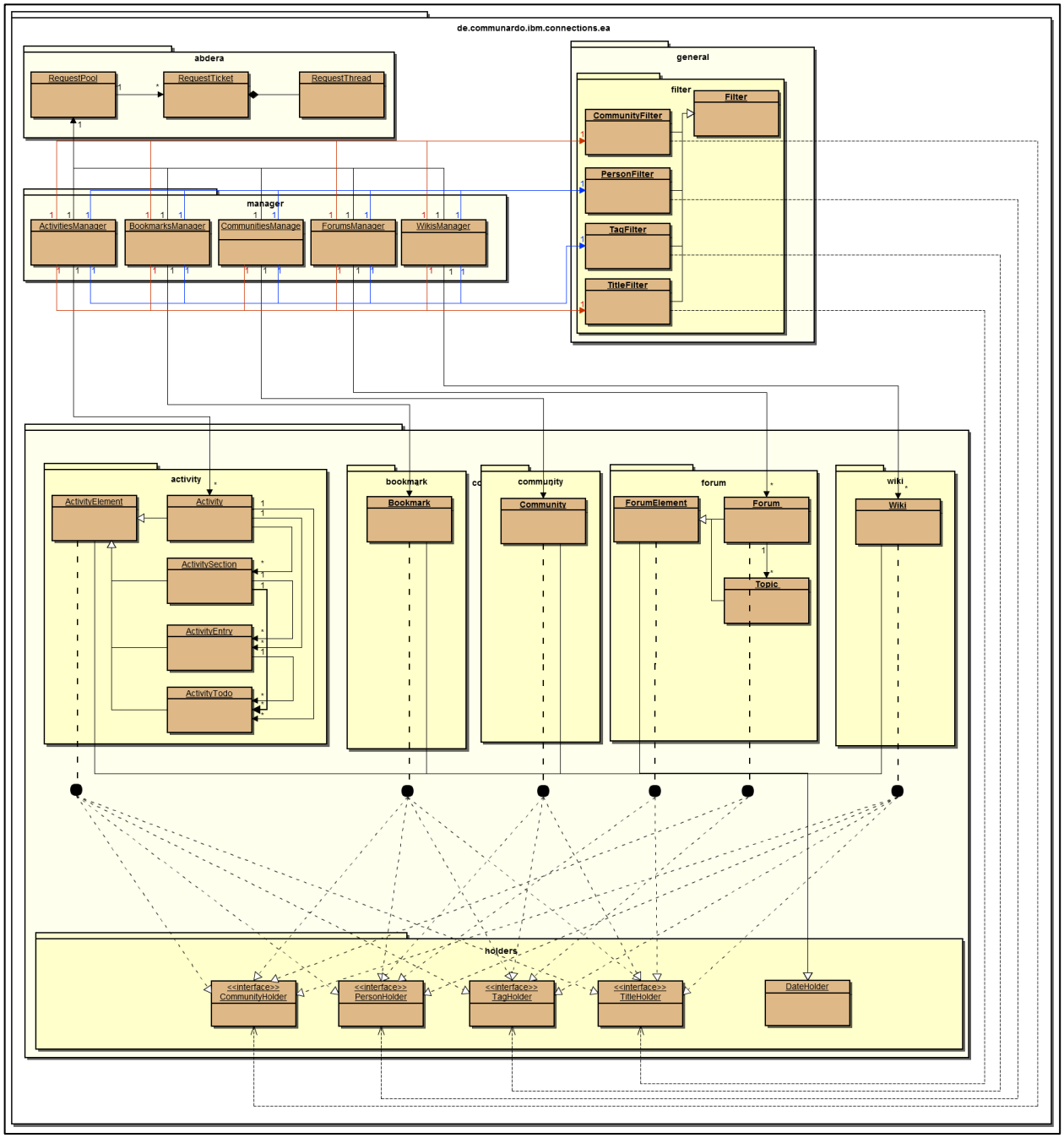


Abbildung 30 Klassendiagramm

4.2.3.2 Package-Hierarchie

Das Programm wurde in mehre Pakete unterteilt, denen jeweils eine besondere Aufgabe zugeteilt ist. Das Grundpaket der Anwendung lautet „de.communardo.ibm.connections.ea“.

abdera

Dieses Paket beinhaltet alle Komponenten um Abfragen das Connections-System zu realisieren. Das Paket beinhaltet 3 Klassen, „RequestPool“, „RequestTicket“ und „RequestThread“ (siehe Abbildung 31). Der „RequestPool“ beinhaltet den Abdera-Client und dessen Konfiguration. Die Konfiguration beinhaltet unter anderem die Cookies, die die Authentifizierung an Connections ermöglich. Ebenfalls verwaltet er die angelegten Tickets. Er bietet die Möglichkeit alle beinhalteten Tickets auf Fertigstellung zu prüfen.

Das „RequestTicket“ ist wiederum ein Container für ein „RequestThread“ und verwaltet ihn damit. Der „RequestThread“ macht die eigentliche Abfrage an Connections.

Das System wurde so gestaltet, um mehrere Abfragen gleichzeitig zu ermöglichen. Dadurch können Abfragen schneller abgearbeitet werden.

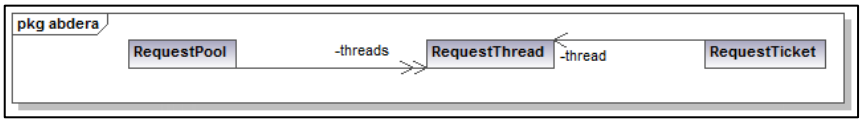


Abbildung 31 Inhalt des "abdera"-Pakets

general

Das Paket „general“ beinhaltet allgemeine Klassen und Objekte, sowie Unterpakete (siehe Abbildung 32). Zu den Klassen und Objekten zählen „Taglist“ und „Person“. „Taglist“ dient dabei als Container für Schlagworte, „Person“ als Abbild von Personen aus Connections. Beide Klassen werden von fast allen Objekten des „components“-Paket verwendet.

Zusätzlich beinhaltet das „general“-Paket auch die Pakete „filter“ und „enums“.

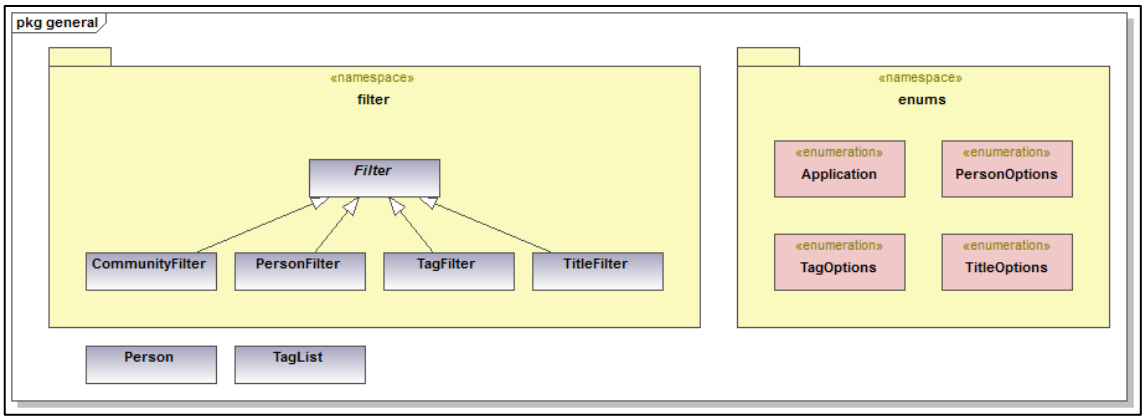


Abbildung 32 Inhalt des "general"-Pakets

general.filter

Dieses Paket beinhaltet alle Filter, die verwendet werden können. Dieses Paket kann beliebig erweitert werden. Alle definierten Filter besitzen eine gemeinsame Super-Klasse „Filter“. Diese Super-Klasse definiert Funktionen, die für den allgemeinen Aufruf der Filter notwendig ist. Dazu zählt die Generierung des Filter-Strings und dessen Rückgabe. Der Filter-String ist ein HTML-Stück, das aus einer Grafik und einem Tooltip besteht (siehe Abbildung 33). Es zeigt im Anzeigemodus welche Filter gewählt wurden.

Tabelle 4 zeigt die definierten Filter-Klassen.

Abbildung 33 Anzeige des Filter-Strings

Tabelle 4 Filter

Filter	Filterdaten	Beschreibung
CommunityFilter	- Community-Name (*) - Community-ID (*) - Option (eine oder beliebig) - Objekte, die “CommunityHolder” implementieren	Der CommunityFilter filtert eine Liste übergebener Daten nach der Community, in der das Objekt enthalten ist. Dies kann nach der übergebenen oder irgendeiner Community-ID erfolgen. Der Community-Name wird nur zur Generierung des „FilterStrings“ benötigt.
PersonFilter	- E-Mail-Adresse des Nutzers - Option (Autor oder Bearbeiter) - Objekte, die „PersonHolder“ implementieren	Der PersonFilter filtert eine Liste übergebener Daten nach dem Autor oder dem letzten Bearbeiter eines Objekts.
TagFilter	- Liste der Schlagwörter - Option (alle oder eins) - Objekte, die „TagHolder“ implementieren	Der TagFilter filtert eine Liste übergebener Daten nach den übergebenen Schlagwörtern. Dabei kann festgelgt werden ob ein Schlagwort oder alle Schlagwörter treffen müssen.

TitleFilter	- Titel - Option (enthält oder exakt) - Objekte, die „TitleHolder“ implementieren	Der TitleFilter filtert eine Liste übergebener Daten nach dem Titel. Dieser Titel muss entweder exakt gleich lauten oder im Titel des Objekts enthalten sein.

*kann auch nicht gegeben sein, wenn die Option „beliebig“ ist

general.enums

Im Paket „enums“ sind Enumerations abgelegt. Diese stellen bis auf „Application“ nur die FilterOptionen dar („PersonOptions“, „TagOptions“, „TitleOptions“). „Application“ nimmt eine Sonderposition ein, da sie teilweise auch Daten für andere Funktionen bereitstellt. Tabelle 5 zeigt die Daten, die in dieser Enumeration enthalten sind. Der Name dient zur Identifikation der Anwendung. Der Link zur Detailsseite, wird auf der globalen Konfigurationsseite benötigt und zeigt, welche Detailseite für welche Anwendung geladen werden muss. Die Anwendungsgruppe definiert die Connections-Anwendung die abgerufen wird. Der TitleKey ist das Schlüsselwort, das in den Language-Properties zur Internationalisierung verwendet wird. Unter Tag-URL ist der Link enthalten, der bei einem Klick auf ein Tag innerhalb der Anwendung aufgerufen wird. Dabei wird zusätzlich das Tag angehangen.

Tabelle 5 Daten der "application"-Enumeration

Name	Link zur Detailseite	Anwen-dungs-gruppe	TitleKey	Tag-URL
PUBLICFORUM	forum/ public.jsp	forum	forums.public	/forums/html/forums? tags=
FOLLOWFORUM	forum/ follow.jsp	forum	forums.my	/forums/html/my? view=following&tags=
MEMBERFORUM	forum/ member.jsp	forum	forums.my	/forums/html/my ?view=member&tags=
OWNERFORUM	forum/ owner.jsp	forum	forums.my	/forums/html/my? view=owner&tags=
PUBLIC BOOKMARKS	bookmark/ public.jsp	bookmark	bookmarks. public	/dogear/html?tag=
MYBOOKMARKS	bookmark/ my.jsp	bookmark	bookmarks.my	/dogear/html?tag=

READERWIKI	wiki/ reader.jsp	wiki	wikis.my	/wikis/home/mywikis? role=reader&tag=
MEMBERWIKI	wiki/ member.jsp	wiki	wikis.my	/wikis/home/mywikis? role=editor&tag=
OWNERWIKI	wiki/ owner.jsp	wiki	wikis.my	/wikis/home/mywikis? role=owner&tag=
PUBLICWIKI	wiki/ public.jsp	wiki	wikis.public	/wikis/home/public?tag=
FOLLOW COMMUNITY	community/ follow.js	community	communities. my.following	/communities/service/html / mycommunities? filterType=following&tag=
MEMBER COMMUNITY	community/ member.jsp	community	communities. my.member	/communities/service/html / mycommunities? filterType=member&tag=
OWNER COMMUNITY	community/ owner.jsp	community	communities. my.owner	/communities/service/html / mycommunities? filterType=owner&tag=
PUBLIC COMMUNITY	community/ public.jsp	community	communities. public	/communities/service/html / allcommunities?tag=
ACTIVITY	activity/ activity.jsp	activity	activity	/activities/service/html/ mainpage #dashboard,myactivities, tag=

manager

Die im Paket „manager“ beinhalteten Klassen verwenden das „abdera“-Paket um Daten aus Connections zu laden und sorgen anschließend dafür, dass die entsprechenden Typen aus dem „components“-Paket erstellt werden (siehe Abbildung 34). Die Manager sorgen also dafür, dass die allgemeinen Rückgabedaten aus der API-Abfrage in eine angepasste Form übertragen werden, die anschließend ausgewertet werden kann.

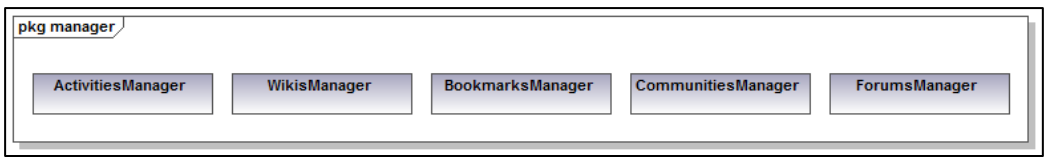


Abbildung 34 Inhalt des "manager"-Pakets

components

Das „components“-Paket beinhaltet die Datenmodelle. Diese sind in mehreren Paketen unterteilt, die jeweils die Daten einer Connections-Anwendung darstellen (siehe Abbildung 35). Eine besondere Rolle spielt dabei das Paket „holders“. In diesem Paket sind einige Schnittstellen realisiert, die für die Filterung benötigt werden. Dies ermöglicht, dass nicht für jede Anwendung ein eigener Filter entworfen werden muss, es muss lediglich das korrekte Interface verwendet werden.

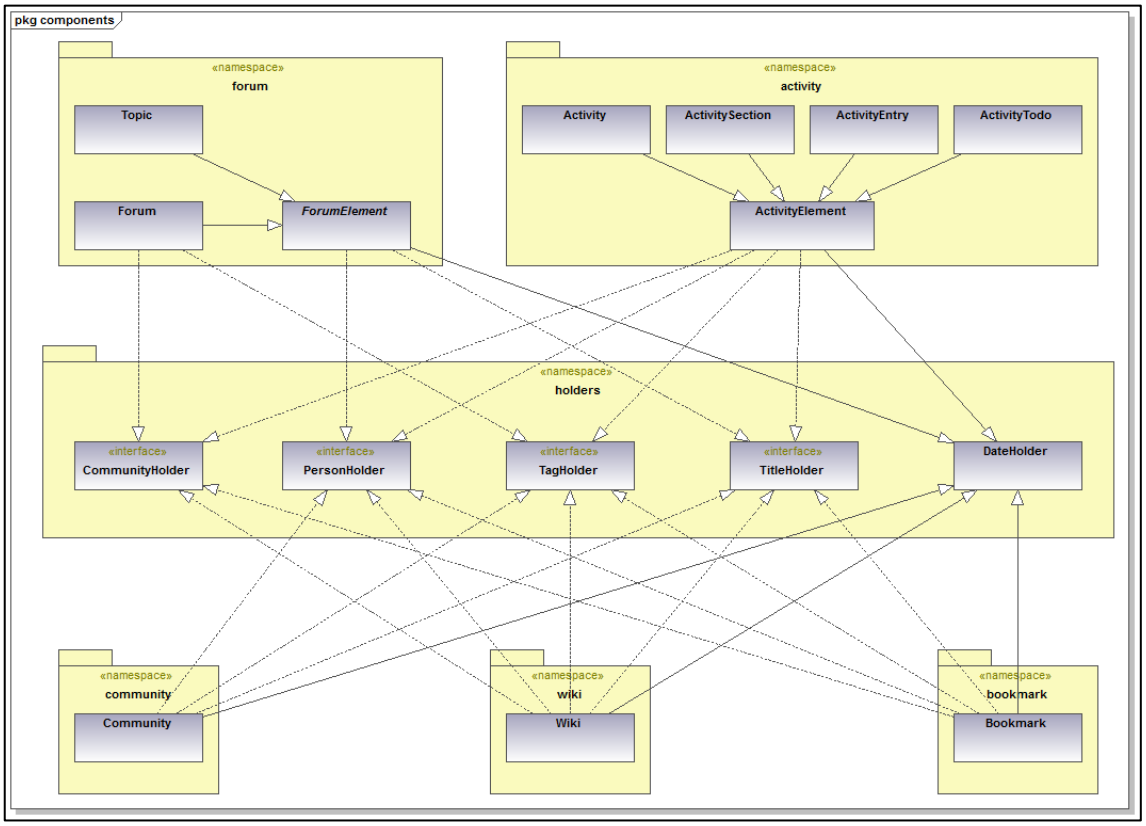


Abbildung 35 Inhalt des "components"-Pakets

actions

Das Paket „actions“ beinhaltet alle Servlets, die für den Programmablauf nötig sind. Alle Klassen in diesem Paket leiten sich von der „Action“-Klasse ab, die sich im „utils“-Paket befindet. Das Paket ist in drei Teile aufgeteilt, die jeweils für einen Bereich zugeordnet sind (siehe Abbildung 36). Im „config“-Paket befinden sich die Servlets die für die Konfiguration notwendig sind. „GlobalConfigAction“ dient als Grundgerüst für die globale Konfigurationsseite, „ApplicationAction“ dient für alle Detailseiten. Das Paket „utils“ beinhaltet Servlets, die für AutoComplete-Funktion benötigt werden. „JsonCommunitesAction“ sorgt dafür, dass die Communitys geladen werden, „JsonTagsAction“ dafür, dass die Tags der jeweils übergebenen Anwendung geladen werden. Beide werden anschließend in der JavaScript-Funktion benötigt. Im „view“-Paket sind die Servlets angelegt, die die Daten für die Darstellung

vorbereiten. Sie sorgen dafür, dass die Daten mithilfe des „manager“-Pakets geladen und anschließend gefiltert werden. Zentraler Zugangspunkt ist dabei die Klasse „DisplayAction“, welche dann, je nach Anwendung, an die eigentliche Display-Klasse weiterleitet.

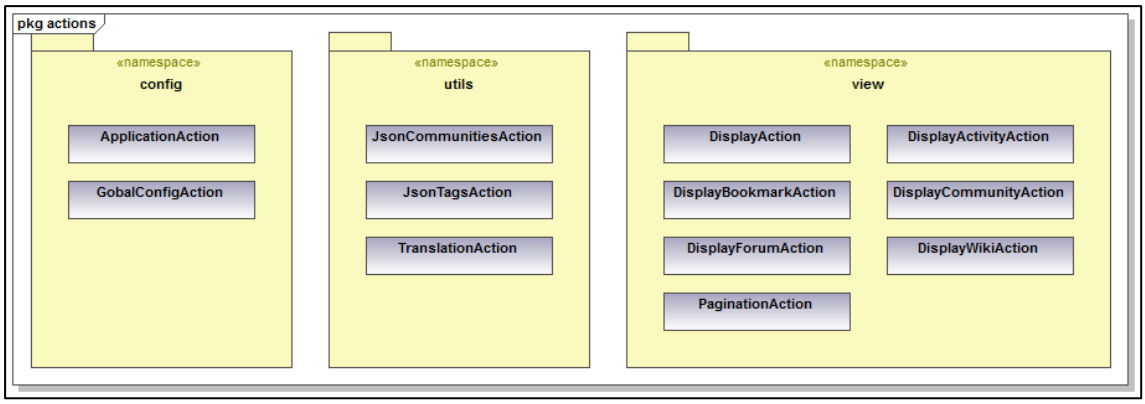


Abbildung 36 Inhalt des "actions"-Pakets

utils

Das Paket „utils“ enthält Klassen, die von mehreren anderen Paketen verwendet werden (siehe Abbildung 37). Die Klasse „Utils“ besteht aus statischen Methoden, die von vielen der „components“-Klassen verwendet werden. Darunter befindet sich zum Beispiel eine Funktion die eine ID aus einer Zeichenkette ausliest. Die „Action“-Klasse dient als Eltern-Klasse für die Servlets im „actions“-Paket.

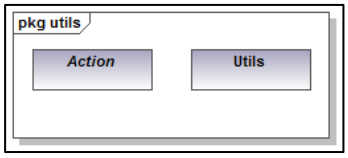


Abbildung 37 Inhalt des "utils"-Pakets

4.2.4 JavaServerPages

4.2.4.1 Aufbau

Die JavaServerPages liegen alle im WebContent-Verzeichnis der Webanwendung im Unterverzeichnis „jsp“. Dort sind die Dateien wieder in Verzeichnis einsortiert, die dem Anwendungsfall entsprechen (siehe Abbildung 38). Die Aufteilung erfolgt in „config“, „display“ und „json“.

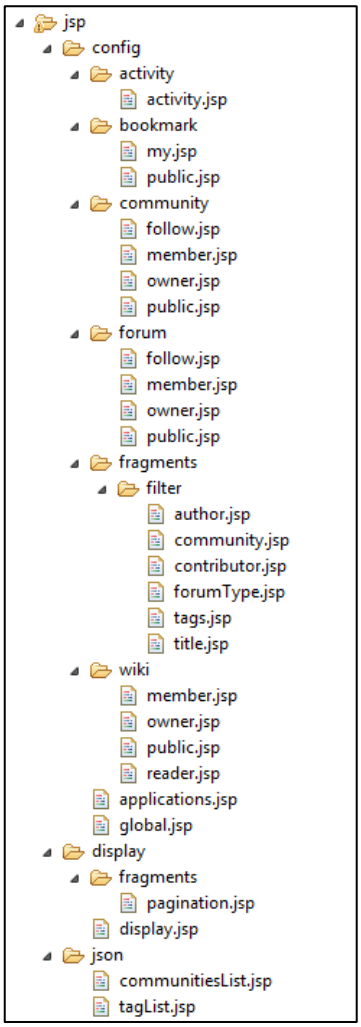


Abbildung 38 JSP-Ordner-Struktur

config

Im „config“-Ordner sind alle JSPs enthalten, die für die Konfiguration benötigt werden. Diese unterteilen sich wiederum in die Ordner der einzelnen Anwendungen. Lediglich die globalen Konfigurationsdateien „global.jsp“ und „application.jap“ liegen direkt im Verzeichnis. Beide sind für die globale Konfigurationsseite zuständig. „global.jsp“ realisiert die Einstellung der maximal anzuzeigenden Elemente, „application.jsp“ übernimmt die Liste der zur Verfügung stehenden Anwendungen.

Die Dateien, die sich innerhalb der Unterordner befinden, beinhalten die Detailseiten der einzelnen Anwendungen. Ausnahme darunter ist der Ordner „fragments“. In diesem Ordner sind Dateien enthalten, die von verschiedenen JSPs der Detailseite eingefügt werden. Sie stellen jeweils die Einstellung einer Filter-Funktion dar (siehe Abbildung 39).

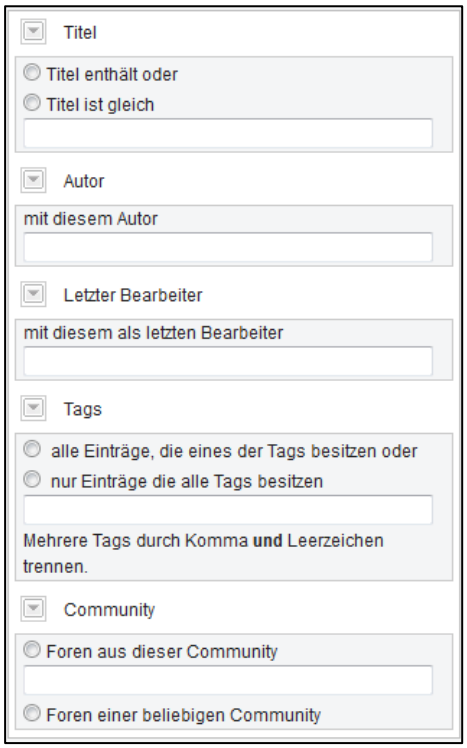


Abbildung 39 Filteroptionen auf einer Detailseite

display

im „display“-Verzeichnis befinden sich die Dateien, die zur Darstellung der Ergebnisse zuständig sind. Die JSP „pagination.jsp“ im Verzeichnis „fragments“ stellt dabei den Seiten-Auswahl- Balken am unteren Ende des Widgets dar.

json

Im Verzeichnis „json“ sind JSPs enthalten die Objekte in JSON darstellen. Diese werden wiederum für die JavaScript-Funktion AutoComplete verwendet, indem sie Community- oder Tag-Daten in einer für JavaScript leicht auswertbaren Liste anordnen. Code-Ausschnitt 6 zeigt das Ergebnis anhand einer Tagliste.

Code-Ausschnitt 6 JSON-Tagliste

```
{
    "tags":[
        "schlagwort",
        "test",
        "versuch",
        "connections",
        "ibm",
        "tag"
    ]
}
```

4.2.4.2 Besonderheiten

Um den Zugriff auf Daten innerhalb der JSP zu vereinfachen wurden Taglibs eingesetzt. Verwendet wurde die JSTL Core und die JSTL FMT („Formatting“). JSTL Core bietet Schleifenkonstrukte und Verzweigungen, die für die Darstellung der Daten notwendig sind. Die JSTL FMT erweitert JSPs mit den Funktionen der Formatierung und Internationalisierung. Dies ermöglicht mithilfe der Language-Properties die mehrsprachige Anzeige.

4.2.4.3 Zusammenspiel mit Servlets

Da JavaServerPages lediglich zur Darstellung verwendet werden, wird auf der Gegenseite jeweils eine Komponente benötigt, die die Daten organisiert und vorbereitet. Diesen Part übernehmen die einzelnen Servlets. Tabelle 6 zeigt welche Servlets welche JSPs verwendet.

Tabelle 6 Verknüpfung von Servlets und JavaServerPages

Servlet	URL-Mapping	Darstellende JSP	Funktion der JSP
Edit-Modus			
GlobalConfigAction	/config/global	/jsp/config/global.jsp	Pagination-einstellung, Buttons
		* /jsp /config/ applications.jsp	Liste der Anwendungen
ApplicationAction	/config/application	/jsp /config/activity/ activity.jsp	Detailseite der Activity-Anwendung
		/jsp /config/ bookmark/*.jsp	Detailseiten der Lesezeichen-Anw.
		/jsp /config/ community/*.jsp	Detailseiten der Community-Anw.
		/jsp /config/ forum/*.jsp	Detailseiten der Forum-Anwendung
		/jsp /config/ wiki/*.jsp	Detailseiten der Wiki-Anwendung
		** /jsp/config/ fragments/*.jsp	Filteroptionen für die Detailseiten
JSON			
JsonCommunitites-Action	/utils/communities	/jsp /json/ communitiesList.jsp	Communitys-Liste im JSON-Format

JsonTagsAction	/utils/tags	/jsp /json/ tagList.jsp	Tag-Liste im JSON-Format
View-Modus			
DisplayAction	/display	*Weiterleitung zu anderen View-Modus-Servlets, je nach Anwendung*	
DisplayActivity-Action	/display/activity	*Weiterleitung zu PaginationAction*	
DisplayBookmark-Action	/display/bookmark	*Weiterleitung zu PaginationAction*	
DisplayCommunity-Action	/display/community	*Weiterleitung zu PaginationAction*	
DisplayForumAction	/display/forum	*Weiterleitung zu PaginationAction*	
DisplayWikiAction	/display/wiki	*Weiterleitung zu PaginationAction*	
PaginationAction	/utils/page	/jsp /display/display.jsp	Hauptanzeigeseite
		** /jsp /display/ fragments/ pagination.jsp	Anzahl der Elemente, Weiter/Zurück-Button (Seiten-Auswahl- Balken)

* Nachladen mittels JavaScript (AJAX) ** Integration in andere JSPs („page incude“)

4.3 Zusammenspiel der Komponenten

4.3.1 Standardablauf im Konfigurationsmodus

Abbildung 40 zeigt den Standardablauf, der im Konfigurationsmodus verwendet wird. Das Diagramm beginnt mit dem Widget in der linken oberen Ecke. Nach dem Laden des Widget wird mittels JavaScript geprüft, ob die Konfiguration vollständig ist, da dies ein Grund ist in den Konfigurationsmodus zu wechseln. Dies kann auch vom Nutzer initialisiert werden. Im Anschluss wird das Servlet „GlobalConfigAction“ geladen, das wiederum dafür sorgt, dass eine JSP geladen wird, die den Button zum Zurücksetzen des Filters und Einstellmöglichkeiten für die maximale Anzahl der Elemente der später anzuzeigenden Seite bereithält. Zusätzlich wird über JavaScript die Liste der zur Verfügung stehenden Anwendungen geladen. Der Nutzer hat anschließend die Möglichkeit eine Anwendung auszuwählen. Dadurch wird das Servlet „ApplicationAction“ geladen, die mithilfe der Enumeration „general.enums.Application“ prüft,

welche JSP im Anschluss für die Detailseite (*) geladen werden muss. Die Detailseite wird währende des Ladens mittels „page include“ mit den definierten Filteroptionen erweitert. Damit die Filteroptionen ordnungsgemäß funktionieren, wird eine Liste der Communitys und der Tags mittels JavaScript geladen. Das Laden der Communitys wird durch das Servlet „JsonCommunitiesAction“ initialisiert. Dieser Servlet lädt einen „manager.CommunitiesManager“. Dieser erstellt einen „abdera.RequestPool“ mit dem wiederum eine Abfrage an das Connections-System umgesetzt werden kann. Die rückgegebenen Daten werden im Anschluss mithilfe einer JSP in die benötigte JSON-Form gebracht, damit sie leichter mit JavaScript ausgewertet werden kann. „JsonTagsAction“ realisiert die Abfrage der Tag-Liste. Da die Tags je nach Anwendung unterschiedlich sind, muss dem AJAX-Request mitgegeben werden, welche Tags benötigt werden. Das Servlet lädt im Anschluss den korrekten Manager (**), der wiederum einen „abdera.RequestPool“ aufbaut, mit dessen Hilfe dann einen Abfrage durchgeführt werden kann. Wenn die Detailseite vollständig geladen wurde, hat der Nutzer die Möglichkeit die Filtereinstellungen vorzunehmen, um die Zahl der späteren Ergebnisse zu beschränken. Diese Auswahl wird nach dem Klick auf den Übernehmen-Button mittels JavaScript gespeichert.

Zu Abbildung 40:

* Zur besseren Darstellung der Abbildung wurden die möglichen Detailseiten zusammengefasst.

Mögliche Detailseiten sind:

- „/jsp /config/activity/activity.jsp“,
- „/jsp /config/bookmark/my.jsp“,
- „/jsp /config/bookmark/public.jsp“,
- „/jsp /config/community/follow.jsp“,
- „/jsp /config/community/member.jsp“,
- „/jsp /config/community/owner.jsp“,
- „/jsp /config/community/public.jsp“,
- „/jsp /config/forum/follow.jsp“,
- „/jsp /config/forum/member.jsp“,
- „/jsp /config/forum/owner.jsp“,
- „/jsp /config/forum/public.jsp“,
- „/jsp /config/wiki/member.jsp“,
- „/jsp /config/wiki/owner.jsp“,
- „/jsp /config/wiki/public.jsp“ und
- „/jsp /config/wiki/reader.jsp“.

** Ebenfalls wurde zur besseren Übersicht die Liste der möglichen Manager gekürzt.

Mögliche Manager sind:

- „manager.ActivitiesManager“,
- „manager.BookmarksManager“
- „manager.CommunitiesManager“,
- „manager.ForumsManager“ und
- „manager.WikisManager“.

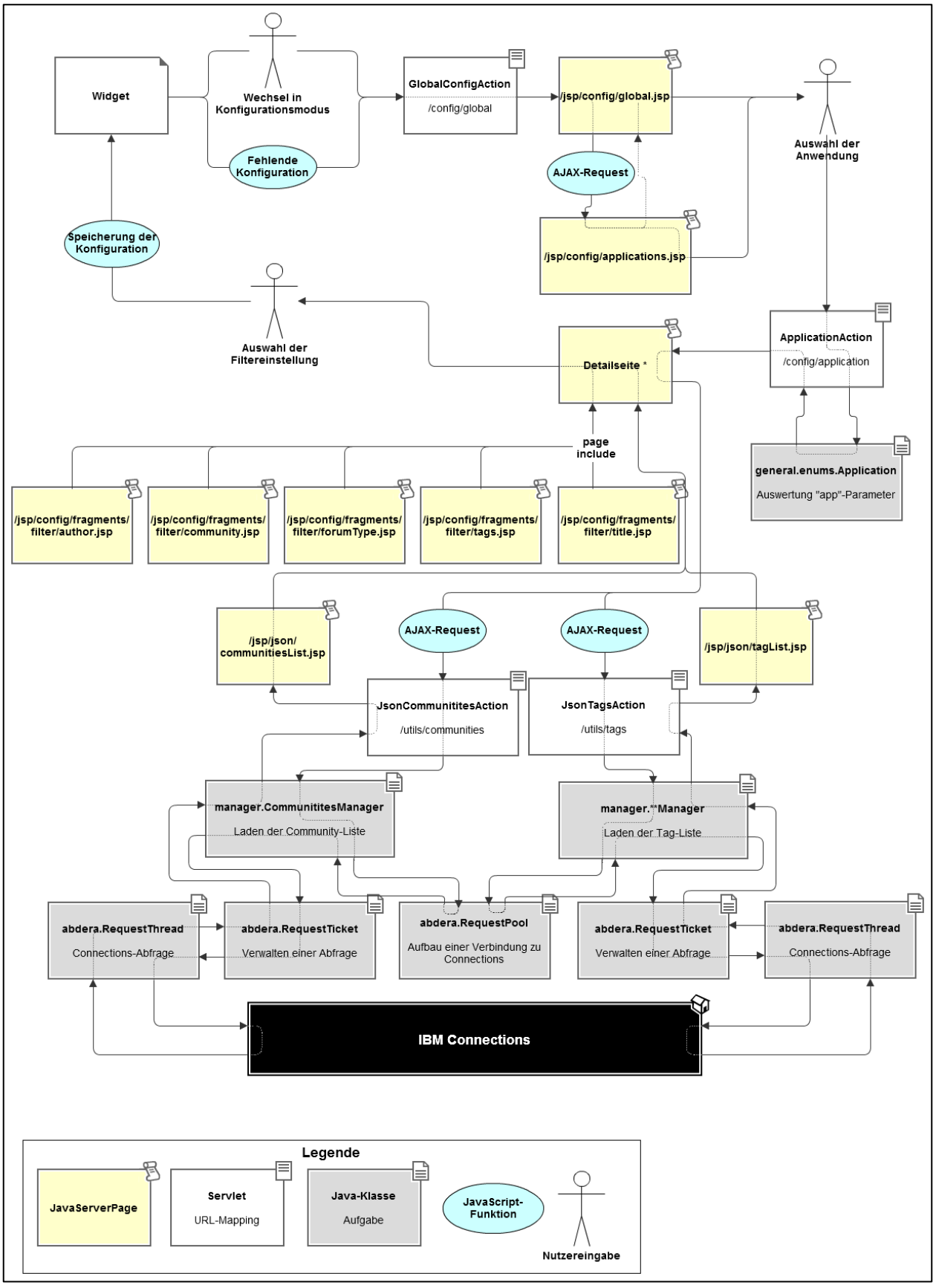


Abbildung 40 Standardablauf im Konfigurationsmodus

4.3.2 Standardablauf im Anzeigemodus

Abbildung 41 zeigt den Standard-Ablauf der Anwendung im Anzeigemodus. Das Diagramm beginnt mit dem Widget in der unteren linken Ecke. Zu Beginn wird mittels JavaScript die Konfiguration geladen und an das Servlet „DisplayAction“ gesendet. Dort wird der „app“-Parameter mithilfe der Enum „general.enums.Application“ ausgewertet. Anhand dieser entscheidet das Servlet an welche spezielle Display-Action-Servlet (*) weitergeleitet werden muss. Jede dieser speziellen Display-Action-Servlets lädt einen speziellen Manager (**), der wiederum mithilfe eines „abdera.RequestPool“ und den dazugehörigen „abdera.RequestTicket“ und „abdera.RequestThread“, das Auslesen der Daten aus Connections übernimmt. Anschließend gibt der Manager die ausgelesenen Daten an eine spezielle „components“-Klasse (***) weiter, die die Daten in Java-Objekte umwandelt. Die generierten Java-Objekte werden anschließend an das ausführende Display-Action-Servlet zurückgegeben. Dieses Servlet sorgt dann dafür, dass die Daten durch die speziellen Filter des Pakets „general.filter“ analysiert und aussortiert werden. Die erhaltene Ergebnismenge wird anschließend an das Servlet „PaginationAction“ weitergegeben, das die Paramater des Seiten-Auswahl- Balken auswertet und die Daten auf die gewünschte Menge begrenzt. Anschließend werden die Daten mithilfe der JSP „/jsp/display/display.jsp“ angezeigt. Diese Seite wird lediglich durch den Seiten-Auswahl- Balken der pagiantion-JSP („/jsp/display/fragments/pagination.jsp“) erweitert. Das Ergebnis wird letztendlich im Widget angezeigt.

Zu Abbildung 41:

Zur besseren Darstellung der Abbildung wurden die möglichen speziellen Display-Action-Servlets (*) zusammengefasst, die Liste der möglichen Manager gekürzt (**) und durch die Vielzahl an „compontents“-Klassen wurde auch diese Anzeige beschränkt (***). Weitere Informationen zeigt Tabelle 7.

Tabelle 7 Fehlende Informationen der Abbildung 41

* Display-Action-Servlets (URL-Mapping)	** Manager-Klasse	*** „components“-Klassen
DisplayActivityAction (/display/activity)	manager.ActivitiesManager	components.activity. Activity components.activity. ActivityElement components.activity. ActivityEntry components.activity. ActivitySection components.activity. ActivityTodo
DisplayBookmarkAction (/display/bookmark)	manager.BookmarksManager	components.bookmark.Bookmark
DisplayCommunityAction (/display/community)	manager.CommunitiesManager	components.community.Community
DisplayForumAction (/display/forum)	manager.ForumsManager	components.forum.Forum components.forum.ForumElement components.forum.Topic
DisplayWikiAction (/display/wiki)	manager.WikisManager	components.wiki.Wiki

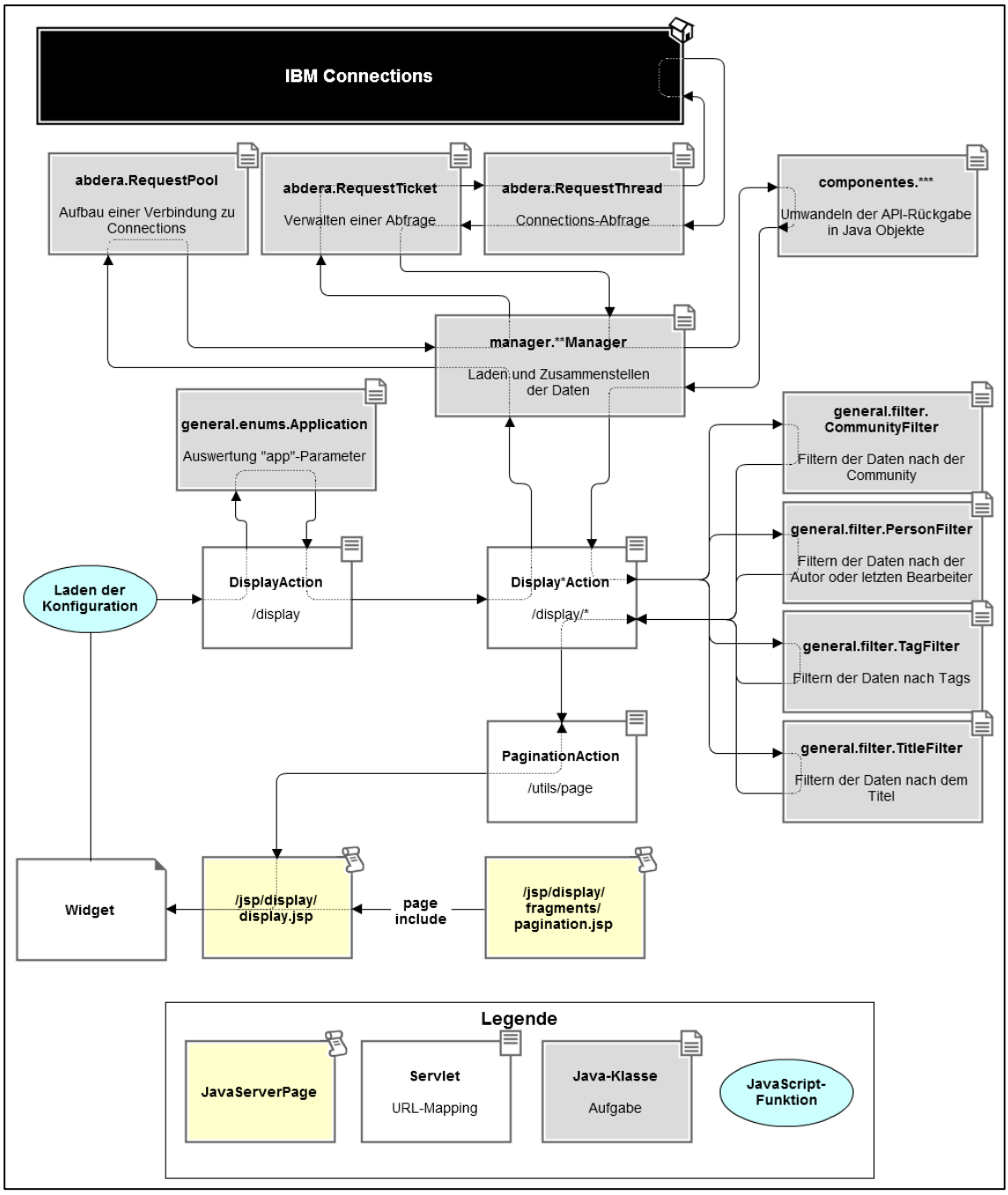


Abbildung 41 Standardablauf im Anzeigemodus

5 Fazit und Ausblick

5.1 Fazit

Ziel der Arbeit war es, dem Nutzer ein Werkzeug zu geben, mit dem er selbst entscheiden kann, welche Inhalte er auf den Überblickseiten angezeigt bekommt. Durch die Umsetzung des Prototyps mit Widgets kann der Nutzer aktiv beeinflussen, was wichtig für ihn ist und letztendlich einen Platz auf seiner Überblickseite findet. Dabei wird er bei der Einstellung mit einer übersichtlichen Konfigurationsseite unterstützt, die ihm hilft seine Filter anzupassen. Der Nutzer hat damit die Möglichkeit nur die Inhalte auf der Seite anzuzeigen, die er für nötig empfindet. Zusätzlich wurden Funktionen hinzugefügt, die Connections bisher nicht bot. So ist es möglich Foren auf der Homepage anzuzeigen. Aufgrund dieser Besonderheit bietet die Anzeige der Foren eine zusätzliche Filterung nach Typen. So ist es möglich untergeordnete Themen aus Foren direkt anzuzeigen (siehe Abbildung 42).

Abbildung 42 Typ-Filter in Foren

Leider konnten in der gegebenen Zeit nicht alle Connections-Anwendungen in dem Widget untergebracht werden. Die Anwendung „Blogs“ konnte aufgrund ihrer ungenauen Definition des Zugangspunktes nicht umgesetzt werden. Es wäre eine Abfrage der Connections-Konfiguration nötig gewesen, dies entsprechend der anderen Anwendungen zu realisieren. Ebenso kann auch die Anwendung „Dateien“ nicht im Widget angezeigt werden. In diesem Fall sind die möglichen Abfragedaten der API zu differenziert um sie in ein einheitliches Schema zu bringen.

5.2 Ausblick

In einer nachfolgenden Version sollten die fehlenden Anwendungen „Blogs“ und „Dateien“ umgesetzt werden, um Nutzern damit den gleichen Mehrwert zu bieten, den es bei den

anderen Anwendungen bereits gibt. Ebenso kann die Anzahl der Filter erweitert werden. Es wäre beispielsweise möglich den Inhalt nach Erstellungsdatum, Beliebtheit oder ähnliches zu filtern. Zusätzlich kann die Sortierung in späteren Versionen eine wichtigere Rolle spielen. Es könnte dem Nutzer die Möglichkeit geboten werden, festzulegen, welche Eigenschaft als Sortiergrundlage dient.

ix Literaturverzeichnis

1. **The Dojo Foundation.** dojo toolkit. [Online] 2011. [Zitat vom: 10. September 2011.] http://dojotoolkit.org/.

2. **Pena (IBM), Ronny A.** IBM Connections wiki. *Lotus Connections iWidget Development Guide.* [Online] 9. März 2010. [Zitat vom: 02. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/development-guide.

3. **Zhong, Qian Jie (Jove), et al., et al.** Developing widgets for IBM Mashup Center 1.0. *IBM developerWorks.* [Online] 19. August 2008. [Zitat vom: 2. September 2011.] http://www.ibm.com/developerworks/lotus/library/mashups-widgets/.

4. **IBM Corp.** Portlets für Business Space-Widgets in WebSphere Portal erstellen. *IBM Websphere Infocenter.* [Online] 28. Juni 2010. [Zitat vom: 5. September 2011.] http://publib.boulder.ibm.com/infocenter/dmndhelp/v6r2mx/index.jsp?topic=/com.ibm.websphere.wbpm.bspace.config.620.doc/doc/tcfg_bsp_portal_portlet.html.

5. **The jQuery Project.** jQuery. [Online] 2010. [Zitat vom: 26. September 2011.] http://jquery.com/.

6. **Oracle Corp.** Class ResourceBundle. *Java Standard Ed. 7 API Docs.* [Online] 2011. [Zitat vom: 25. September 2011.] http://download.oracle.com/javase/7/docs/api/java/util/ResourceBundle.html.

7. **IBM Corp.** Lotus Quickr. *IBM Software.* [Online] 2011. [Zitat vom: 28. September 2011.] http://www-01.ibm.com/software/de/lotus/quickr/.

8. —. WebSphere Application Server. *IBM Software.* [Online] 2011. [Zitat vom: 10. September 2011.] http://www-142.ibm.com/software/products/de/de/appserv-was.

9. **Vodafone D2 GmbH.** Glossar. *Vodafone Innovation.* [Online] 2011. [Zitat vom: 30. September 2011.] http://www.vodafone.de/unternehmen/innovationen/glossar.html#ID172255.

10. **IBM Corp.** IBM Connections. *IBM Software.* [Online] 2011. [Zitat vom: 28. September 2011.] http://www-01.ibm.com/software/lotus/products/connections/.

11. —. Connections 3.0.1 System Requirements. *IBM Support Portal.* [Online] 30. August 2011. [Zitat vom: 10. September 2011.] https://www-304.ibm.com/support/docview.wss?uid=swg27021342.

12. **Sun Microsystems.** *Java™ Platform, Enterprise Edition (Java EE) Specification, v5.* 28. April 2006.

13. **Zang (IBM), Hao.** Installing Lotus Connections 3.0. *IBM Connections wiki.* [Online] 28. April 2011. [Zitat vom: 14. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Installing_Lotus_Connections_3.0_lc3.

14. **IBM Corp.** Troubleshooting a database connection. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 16. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Troubleshooting_a_database_connection_ic301#.

15. —. Communities. *IBM Connections .* [Online] 2011. [Zitat vom: 29. September 2011.] http://www-01.ibm.com/software/lotus/products/connections/communities.html.

16. **Barrett (IBM), Paddy.** What is an Ideation Blog? *IBM Connections wiki.* [Online] 5. Mai 2011. [Zitat vom: 29. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/What_is_an_Ideation_Blog_ic301.

17. **English (IBM), Jason.** Adding a media gallery to your community. *IBM Connections wiki.* [Online] 19. September 2011. [Zitat vom: 29. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Adding_a_media_gallery_to_your_community_ic301.

18. **IBM Corp.** Using an Ideation Blog. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 23. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Using_an_Ideation_Blog_ic301.

19. **IBM Cor.** Welcome to wikis. *IBM Connections wiki.* [Online] 2. Februar 2011. [Zitat vom: 4. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Welcome_to_wikis_lc3.

20. **IBM Corp.** Welcome to Forums. *IBM Connections wiki.* [Online] 2. Februar 2011. [Zitat vom: 4. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Welcome_to_Forums_lc3.

21. —. What's new in this release? *IBM Connections wiki.* [Online] 2. Februar 2011. [Zitat vom: 4. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Whats_new_in_this_release_lc3.

22. —. Neuerungen in diesem Release. *IBM Lotus Connections 2.5.* [Online] 29. Oktober 2010. [Zitat vom: 29. September 2011.] http://publib.boulder.ibm.com/infocenter/ltscnnct/v2r0/index.jsp?topic=/com.ibm.connections.25.help/i_ovr_r_whats_new.html.

23. —. Welcome to Profiles. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 23. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Welcome_to_Profiles_ic301.

24. —. Home page widgets. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 29. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Home_page_widgets_ic301.

25. —. Communities widgets. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 29. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Communities_widgets_ic301.

26. —. Developing. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 29. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Developing_ic301.

27. —. Event SPI. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 29. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Event_SPI_ic301.

28. —. Seedlist SPI. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 29. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Seedlist_SPI_ic301.

29. —. IBM Connections API overview. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 29. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/IBM_Connections_API_overview_ic301.

30. **The Apache Software Foundation.** Apache Abdera. [Online] 2010. [Zitat vom: 23. September 2011.] http://abdera.apache.org/.

31. **IBM Corp.** Navigating Activities resources. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 25. August 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Navigating_Activities_resources_ic301.

32. —. Navigating Blogs subscription API resources. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 23. August 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Navigating_Blogs_subscription_API_resources_ic301.

33. —. Navigating Bookmarks resources. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 23. August 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Navigating_Bookmarks_resources_ic301.

34. —. Navigating Communities resources. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 26. August 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Navigating_Communities_resources_ic301.

35. —. Getting the My Communities feed. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 23. August 2011.]

36. —. Retrieving the Files service document. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 25. August 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Retrieving_the_Files_service_document_ic301.

37. —. Navigating Forums resources. *IBM Connections wiki.* [Online] 6. April 2011. [Zitat vom: 25. August 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Navigating_Forums_resources_ic301.

38. —. iWidget Specification v1.0. [Online] 28. August 2008. [Zitat vom: 29. September 2011.] http://www-10.lotus.com/ldd/mashupswiki.nsf/dx/iwidget-spec-v1.0.pdf/$file/iwidget-spec-v1.0.pdf.

39. **Estrada (IBM), Tony.** Integrating the Profiles business card. *IBM Connections wiki.* [Online] 22. Februar 2011. [Zitat vom: 21. September 2011.] http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Integrating_the_Profiles_business_card_lc3.

40. What's new in this release? *IBM Connections wiki.* [Online] 2. Februar 2011. http://www-10.lotus.com/ldd/lcwiki.nsf/dx/Whats_new_in_this_release_lc3.

x Verwendete Hilfsmittel

Als Test- und Entwicklungssystem wurde verwendet:

- IBM Connections 3.0.1
- IBM WebSphere Application Server 7.0.0.11
- IBM DB2 V9.5 Enterprise
- IBM Tivoli Directory Server 6.2
- Windows Server 2008

Zur Entwicklung wurde verwendet:

- IBM Rational Application Developer 8.0
- Eclipse Helios SR1

Zur Erstellung der Grafiken wurde folgende Software verwendet:

- Gliffy - Online Diagram Software and Flowchart Software (http://www.gliffy.com)
    - Abbildung 1 bis 13, 17, 20, 30, 40, 41
- Altova UModel 2011 Enterprise Edition
    - Abbildung 31, 32, 34 bis 37
- Abbildungen 21 bis 19, 33, 38, 39 und 42 zeigen Screenshots des entwickelten Prototyps.

xi Danksagung

Ich möchte mich an dieser Stelle bei denen bedanken, die mich bei der Erstellung meiner Bachelorarbeit unterstützt haben. Besonderen Dank gilt Prof. Wilfried Schubert, der mich ausgezeichnet betreut hat und immer gute Ratschläge für die Bearbeitung parat hatte. Ebenso möchte ich mich besonders bei Tino Schmidt bedanken, der mich beim Finden des spannenden Themas und auch bei der Bearbeitung und Umsetzung unterstützt hat. Dank gilt auch der Communardo Software GmbH die mich finanziell und auf der technischen Seite unterstützt hat, indem sie mir Arbeitsmaterial und Lizenzen für Software zur Verfügung gestellt haben. Weiterhin möchte ich Alexander Stephan, Markus Weigelt und anderen Mitarbeitern der Communardo Software GmbH danken, die mir jederzeit die Antworten auf Implementierungsfragen gegeben haben.

Ich möchte auch meiner Familie danken, die mich mit allen Mitteln unterstützt hat und besonders meiner Verlobten, die sich liebevoll um unseren Sohn kümmerte, wenn ich leider keine Zeit dafür hatte.

xii Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt habe.

Stellen, die wörtlich oder sinngemäß aus Quellen entnommen wurden, sind als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Mittweida, den 04.10.2011

Andreas Rieger